UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

IA-GPS: Uma Ferramenta Baseada em Algoritmos de Busca para Alocação de Equipes em Projetos de Software

Trabalho de Conclusão de Curso

Werthen de Castro Santos

São Cristóvão – Sergipe

2019

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Werthen de Castro Santos

IA-GPS: Uma Ferramenta Baseada em Algoritmos de Busca para Alocação de Equipes em Projetos de Software

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador(a): Leila Maciel de Almeida e Silva
Coorientador(a): André Britto de Carvalho

São Cristóvão – Sergipe

2019

Dedico esse trabalho a minha família, professores e amigos que acompanharam nessa jornada da graduação, que tanto me aconselharam e me apoiaram até o final.

Agradecimentos

A jornada da graduação normalmente é bem diferente do que imaginamos antes de entrar na universidade, cheia de desafios e oportunidades das quais algumas eu perdi e outras aproveitei bastante. Conheci um mundo totalmente diferente e incrível.

Em especial, devo agradecer aos professores que mais me envolvi durante a graduação: Profa. Leila, Prof. André, Prof. Bruno e Profa. Débora. A professora Leila, foi minha professora que mais tive contato, foi minha orientadora da iniciação cientifica e orientadora deste trabalho, além de minha professora da matéria de Grafos. Foi também a quem dei mais trabalho, já que sempre corrigiu meus erros e me orientou durante boa parte da graduação.

O professor André foi meu coorientador na iniciação cientifica, onde me ajudou bastante no entendimento do problema e do algoritmo utilizado, além de ser também coorientador deste trabalho e meu professor de Banco de Dados. O professor Bruno foi meu orientador no PRODAP por dois anos, onde tive boas experiências e lutamos para realizar um projeto que possibilitava a redução custos para UFS, utilizando impressora 3D e também teve uma ótima didática quando foi meu professor de Análise de Algoritmos. A professora Débora que foi minha professora de Desenvolvimento de Software 1 e 3, tendo passado sua experiência de mercado.

Por fim, agradeço aos meus companheiros de graduação, com os quais compartilhei a alegria e o sofrimento durante as matérias que fizemos juntos, os finais de semana que passamos estudando para provas e e também aqueles que saímos para nos divertirmos, as brincadeiras de corredor e é claro, o tempo diário que passamos juntos no RESUN. Particularmente agradeço, Jackson Tavares, Davi Silva, Antônio Carlos, Ítalo Jorge, Paulo de Brito, João Cruz, Marina Vivas, Felipe Pereira, Gabriel Leite, Itamar Souza, Elias Rabelo, Diogo de Lima, Jonas Bastos, Paulo Roberto, Carlos Tadeu, Igor Figueiredo, Murilo Almeida, Pedro Henrique, José Joaquim e tantos outros com os quais convivemos nestes anos de graduação.

"Um livro é a prova de que os homens são capazes de fazer magia."
(Carl Sagan)

Resumo

O planejamento de um grande projeto envolve muitas variáveis, incertezas e objetivos conflitantes, sendo assim uma tarefa muito complexa. Dado que o processo manual pode incorrer em soluções de baixa qualidade, recentemente alguns trabalhos propuseram a utilização de algoritmos de Inteligência Artificial, em particular, meta-heurísticas, para ajudar o gerente de projeto de software na tarefa de planejamento. Esta forma de modelagem insere-se no âmbito da Engenharia de Software Baseada em Busca e o problema de planejamento tratado neste trabalho é conhecido como Problema do Escalonamento e Atribuição de Tarefas em Projetos de Software, o qual objetiva alocar funcionários a tarefas, de forma a otimizar o custo e a duração do projeto. Este trabalho apresenta a IA-GPS, uma aplicação integrada ao *framework* jMetal que simplifica a utilização de algoritmos heurísticos, provendo uma interface mais amigável para os engenheiros de software analisarem as possíveis soluções geradas para o problema. A IA-GPS fornece aos gerentes possíveis alocações de funcionários a tarefas e cronogramas, os quais podem ser visualizados de diversas formas. A aplicação foi desenvolvida em Java e foi validada com estudos de caso sintéticos, extraídos da literatura.

Palavras-chave: Projeto de software, meta-heurística, otimização, escalonamento, multiobjetivo.

Abstract

The planning of a large project involves many variables, uncertainties and conflicting objectives, being a very complex task. Given that the manual process may lead to low quality solutions, recently some papers have proposed the use of Artificial Intelligence algorithms, in particular metaheuristics, to help the software project manager in the planning task. This form of modeling falls within the scope of Search-Based Software Engineering and the planning problem addressed in this work is known as Software Project Scheduling Problem whose goal is to allocate employees to project tasks, in order to optimize project cost and duration. This work presents IA-GPS, an application integrated with jMetal that simplifies the use of heuristic algorithms, providing a more user-friendly interface for software engineers to analyse the possible solutions for the problem, generated by the algorithms. IA-GPS provides managers with possible employee allocations for tasks and timelines, which can be viewed in a variety of ways. The application was developed in Java and was validated with synthetic case studies, extracted from the literature.

Keywords: software project, metaheuristics, optimizes, scheduling, multiobjective.

Lista de ilustrações

Figura 1 – Grafo direcionado acíclico. 16
Figura 2 – Matriz de dedicação de um projeto. 17
Figura 3 – Gráfico com a fronteira de Pareto. 19
Figura 4 – Transformação de matriz em vetor. 20
Figura 5 – Gráfico de Gantt 21
Figura 6 – Representação do cálculo do hipervolume. 22
Figura 7 – Diagrama de funcionamento do desenvolvimento incremental. 23
Figura 8 – Notação UML - diagrama casos de uso. 24
Figura 9 – Notação UML - diagrama de atividades 24
Figura 10 – Diagrama de classes - componentes de uma classe. 25
Figura 11 – Diagrama de classes - multiciplidade. 25

Figura 12 – Diagrama de casos de uso da aplicação - parte 1. 31
Figura 13 – Diagrama de casos de uso da aplicação - parte 2. 32
Figura 14 – Descrição de caso de uso - Cadastrar Projeto 33
Figura 15 – Diagrama de atividades 35
Figura 16 – Fragmento do diagrama de classes do projeto lógico 36
Figura 17 – Tela do usuário 38
Figura 18 – Tela de criação de um novo projeto 38
Figura 19 – Telas de cadastro de habilidades, funcionários e tarefas 39
Figura 20 – Tela de soluções e dados do projeto. 40
Figura 21 – Tela gráfico de Gantt. 41
Figura 22 – Tela de matriz de dedicação 41
Figura 23 – Tela mapa de distribuição de dedicação 42
Figura 24 – Tela com grafo de dependência de tarefas 43

Figura 25 – Descrição de caso de uso - Cadastrar os dados do Projeto 50
Figura 26 – Descrição de caso de uso - Visualizar o diagrama de Gantt. 51
Figura 27 – Descrição de caso de uso - Visualizar o grafo de dependências. 51
Figura 28 – Descrição de caso de uso - Visualizar a matriz de dedicação. 52
Figura 29 – Descrição de caso de uso - Visualizar o mapa de alocação de recursos. . . . 52

Lista de tabelas

Tabela 1 – Variáveis de um funcionário do SPSP 16
Tabela 2 – Variáveis de uma tarefa no SPSP 16
Tabela 3 – Características das ferramentas. 28

Tabela 4 – Requisitos funcionais da aplicação 30
Tabela 5 – Requisitos não funcionais da aplicação 30

Lista de abreviaturas e siglas

SBSE	*Search-Based Software Engineering*
SPSP	*Software Project Scheduling Problem*
SMPSO	*Speed Constrained Multi-objective PSO*
PERT	*Program Evaluation and Review Technique*
CPM	*Critical Path Method*
RCPSP	*Resource-Constrained Project Scheduling Problem*
TCC	Trabalho de Conclusão de Curso
API	*Application Programming Interface*
IDE	*Integrated Development Environment*
MVC	*Model-View-Controller*
IA	Inteligência Artificial
DSPSP	*Dynamic Software Project Scheduling Problem*
UML	*Unified Modeling Language*
MD	Matriz de Dedicação
GDA	Grafo Direcionado Acíclico
PSO	*Particle Swarm Optimization*
IA-GPS	Interface de Apoio à Gerentes de Projetos de Software
NSGA-II	*Nondominated Sorting Genetic Algorithm II*
PAES	*Pareto Archived Evolution Strategy*

Sumário

1 Introdução 11
1.1 Objetivos 12
1.1.1 Objetivo Geral 12
1.1.2 Objetivos Específicos 12
1.2 Metodologia 13
1.3 Estrutura do Documento 13

2 Fundamentação Teórica 15
2.1 O Problema do Escalonamento e Atribuição de Tarefas em Projetos de Software. 15
2.1.1 Algoritmos Heurísticos para o SPSP. 18
2.1.2 Hipervolume. 21
2.2 Metodologia de Desenvolvimento de Software Utilizada no Projeto. 22
2.3 A Linguagem UML. 23
2.4 A Arquitetura MVC 26
2.5 Ferramentas e linguagens utilizadas no desenvolvimento do projeto. 26
2.6 Trabalhos Relacionados 27

3 A Ferramenta IA-GPS 29
3.1 Levantamento de Requisitos 29
3.1.1 Requisitos funcionais e não funcionais 29
3.1.2 Diagrama de Casos de Uso 30
3.2 Projeto da ferramenta 34
3.2.1 Diagrama de Atividades 34
3.3 Diagrama de Classes 35
3.4 Integração com o jMetal 37
3.5 Principais funcionalidades da ferramenta 37
3.6 Validação da Ferramenta 42

4 Conclusões 44

Referências 46

Apêndices 49

APÊNDICE A Casos de uso comentados 50

1 Introdução

Define-se um projeto como um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo (PMI, 2014). Um projeto pode ser a construção de um prédio, um plano de socorro para vítimas de um desastre natural ou o desenvolvimento de um software, por exemplo. A área de gerenciamento de projeto se dedica a aplicação do conhecimento, habilidades, ferramentas e técnicas nas atividades de planejamento e monitoramento do desenvolvimento do projeto, visando assegurar que os requisitos sejam atendidos.

Em particular, projetos de desenvolvimento de software não são simples. Empresas do ramo enfrentam grandes dificuldades que fazem com que seus projetos atrasem, custem mais do que o planejado e até mesmo sejam cancelados. Um estudo do Standish Group (2014) mostra que 31,1% dos projetos de software são cancelados e 52,7% custam mais do que o valor inicial projetado, gerando prejuízos enormes tanto devido ao custo do projeto, como também às oportunidades de negócios perdidas, que nem sempre é possível mensurar precisamente. Fazer um planejamento de um projeto de software é complexo pois envolve muitas variáveis, objetivos conflitantes e incertezas. Esta complexidade dificulta sobremaneira o trabalho de gerentes de projetos.

Assim, a automação da tarefa de planejamento, mesmo que parcial, se torna necessária, para evitar possíveis erros decorridos da forma manual. Existem algumas ferramentas para prover apoio automático à tarefa de planejamento como MS Project, que utiliza a Técnica de Avaliação e Revisão de Projetos (em inglês, *Program Evaluation and Review Technique* (PERT)) e Método do Caminho Crítico (em inglês, *Critical Path Method* (CPM)) (MARTINS; LAUGENI, 2005). Dado que o planejamento de software envolve objetivos conflitantes, como custo e duração do projeto, por exemplo, trabalhos recentes aplicaram algoritmos de otimização para encontrar um balanceamento adequado das variáveis envolvidas no planejamento (HARMAN; MANSOURI; ZHANG, 2009) e (FERRUCCI; HARMAN; SARRO, 2014). Estes trabalhos inserem-se no âmbito da Engenharia de Software Baseado em Busca (em inglês, *Search Based Software*

Engineering (SBSE)).

Dentre os problemas de gerenciamento de projetos que podem ser resolvidos no contexto de SBSE, destaca-se o Problema do Escalonamento e Atribuição de Tarefas em Projetos de Software (em inglês, *Software Project Scheduling Problem* (SPSP)), que consiste na alocação de empregados a tarefas de forma a minimizar a duração e custo do projeto (ALBA; CHICANO, 2007). Cada empregado possui um conjunto de habilidades e um tempo máximo de dedicação ao projeto. Cada tarefa requer um conjunto habilidades de específicas para ser desenvolvida.

Os trabalhos para o SPSP no contexto de SBSE (XIAO; AO; TANG, 2013) e (XIAO; GAO; HUANG, 2015) investigaram, na sua vasta maioria, a adequação de um dado algoritmo para encontrar soluções para o problema. No entanto, poucos trabalhos propuseram uma ferramenta que facilite a utilização destes algoritmos pelo engenheiro de software. Exemplo de softwares nesta direção incluem o RESCON (DEBLAERE; DEMEULEMEESTER; HERROELEN, 2011) e PpcProject (SALAS-MORERA et al., 2013), ferramentas educacionais utilizadas em universidades. Além destas, Stylianou, Gerasimou e Andreou (2012) propuseram um protótipo de uma ferramenta que utiliza algoritmos de Inteligência Artificial (IA) denominada IntelliSPM. A seção 2.6 fornece uma visão mais detalhada dos trabalhos relacionados.

Nota-se, entretanto, que as empresas não costumam utilizar técnicas de IA para resolver o problema de escalonamento de tarefas de software. Este fato deve-se à dificuldade do gerente em aplicar os algoritmos envolvidos sem um suporte adequado, já que a parametrização dos mesmos pode não ser facilmente realizada. Além disso, há o desconhecimento dos benefícios do uso de tais técnicas. Dentro deste contexto, supõe-se que o desenvolvimento de uma ferramenta que proveja mecanismos que facilitem o uso de algoritmos de otimização seja de valoroso interesse.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo deste trabalho é desenvolver uma aplicação que funcione como uma interface, com o objetivo de permitir que gerentes de projetos de software utilizem algoritmos heurísticos para solucionar o Problema do Escalonamento e Atribuição de Tarefas em Projetos de Software de forma mais amigável.

1.1.2 Objetivos Específicos

Alguns objetivos específicos são necessários para atingir o objetivo principal. Esses objetivos são definidos a seguir:

1. Definição das características da aplicação com base nas necessidades dos gerentes de projetos de software, extraídas de trabalhos da literatura;

2. Projeto e implementação da aplicação;
3. Integração da aplicação com o *framework* de otimização jMetal;
4. Realização de testes do software;
5. Análise dos resultados.

1.2 Metodologia

Para o desenvolvimento da aplicação proposta, algumas atividades foram necessárias:

1. **Estudos dos conceitos gerais** sobre os principais tópicos necessários para o desenvolvimento deste trabalho, como meta-heurísticas, métricas de avaliação, metodologias de desenvolvimento de software, *frameworks* para utilizar meta-heurísticas e o problema de escalonamento;
2. **Estudo aprofundado** do Problema do Escalonamento e Atribuição de Tarefas em Projetos de Software e ferramentas que permitam o gerenciamento das tarefas do projeto, com a utilização de meta-heurísticas. no contexto de SBSE;
3. **Levantamento de requisitos** baseado em artigos de SPSP, no contexto de SBSE (ALBA; CHICANO, 2007), (XIAO; AO; TANG, 2013) e (XIAO; GAO; HUANG, 2015).
4. **Elaboração do diagrama de casos de uso**, o qual expressa em forma diagramática os requisitos funcionais do sistema.;
5. **Validação dos casos de uso da ferramenta** através de discussões e apresentação dos casos de uso a (professores e alunos da pós-graduação e graduação) que possuem experiência com o problema do SPSP;
6. **Projeto e implementação da ferramenta** segundo as fases da metodologia de desenvolvimento incremental, e usando a linguagem Java;
7. **Validação da ferramenta** usando uma bateria de experimentos com dados sintéticos;
8. **Integração da aplicação com o o jMetal** de forma que a aplicação seja de fácil uso;
9. **Escrita e defesa desta monografia.**

1.3 Estrutura do Documento

Este trabalho possui a organização descrita a seguir. O capítulo 2 descreve a fundamentação teórica, que é a base conceitual deste trabalho, apresentando conceitos sobre o problema

estudado, as meta-heurísticas, e métodos de desenvolvimento de software. O capítulo 3 descreve a ferramenta proposta e o capítulo 4 tece algumas considerações finais e direções de trabalhos futuros.

2

Fundamentação Teórica

Para poder realizar a proposta deste trabalho, é necessário primeiro entender o Problema do Escalonamento e Atribuição de Tarefas em Projetos de Software, como este é modelado e solucionado. Além disso, descreve-se brevemente alguns algoritmos que podem ser usados em problemas multiobjetivos e como suas soluções são avaliadas. São apresentados também conceitos necessários para o desenvolvimento da aplicação proposta, a metodologia de desenvolvimento da aplicação e as ferramentas necessárias para sua implementação

2.1 O Problema do Escalonamento e Atribuição de Tarefas em Projetos de Software.

O Problema do Escalonamento e Atribuição de Tarefas em Projetos de Software (ALBA; CHICANO, 2007) considera um conjunto de tarefas e um conjunto de funcionários que se dedicarão a executá-las. Cada tarefa possui um custo associado, que pode variar de acordo com as características da tarefa e um conjunto de habilidades necessárias para concluí-la. As tarefas devem ser realizadas numa dada ordem, pois pode haver dependência entre elas. Cada funcionário possui um salário, um conjunto de habilidades e um tempo máximo de dedicação a um projeto, podendo executar várias tarefas e trabalhar em mais de um projeto ao longo de um dia.

Em SPSP o conjunto de funcionários é representado por E, n representa a quantidade de funcionários disponíveis para realizar as tarefas de um dado projeto e S o conjunto de habilidades necessárias para concluir as tarefas. A Tabela 1 sumariza as variáveis relativas a um funcionário no SPSP.

O conjunto de tarefas T de tamanho m, tamanho m, em que cada tarefa requer um conjunto de habilidades para serem executadas. A Tabela 2 sumariza as variáveis relativas a uma tarefa no SPSP. Estas tarefas determinam o fluxo do projeto, podendo haver dependência

Tabela 1 – Variáveis de um funcionário do SPSP

Variável	Descrição
e_i	Um funcionário no projeto
e_i^{maximo}	Dedicação máxima de um funcionário e_i
$e_i^{salario}$	Salário do funcionário e_i
$e_i^{habilidades}$	Habilidades que o funcionário e_i possui

Fonte: Próprio autor.

entre elas. Assim, uma tarefa só pode começar se a tarefas predecessoras tiveram terminado. Essa característica permite modelar a relação entre tarefas como um Grafo Direcionado Acíclico (GDA). Assim, pelo menos uma tarefa não possui predecessor, sendo a mesma um vértice no grafo com grau de entrada igual a zero e esta é a primeira tarefa que será executada.

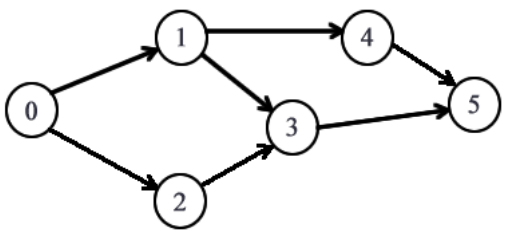
Tabela 2 – Variáveis de uma tarefa no SPSP

Variável	Descrição
t_j	Uma tarefa do projeto
t_j^{custo}	Custo de uma tarefa t_j
$t_j^{esforco}$	Carga de trabalho para realizar t_j medidos em pessoa-mês
dep_j	Conjunto de dependências de uma tarefa t_j
$t_j^{habilidade}$	Habilidades requeridas por uma tarefa t_j
t_j^{dur}	A duração de uma tarefa t_j
t_j^{inicio}	O momento que inicia uma tarefa t_j
t_j^{fim}	O momento que termina uma tarefa t_j

Fonte: Próprio autor.

O GDA é formado por um conjunto de vértices, T, representados pelas tarefas, e um conjunto de aresta, A, que indica a relação de precedência entre as tarefas. Seja uma aresta $(t_i, t_j) \in A$, então a tarefa t_j só pode ser iniciado após t_i ser finalizada. A Figura 1 ilustra um GDA.

Figura 1 – Grafo direcionado acíclico.



Fonte: Próprio autor.

Uma solução para SPSP é representada na forma de uma matriz, denominada de Matriz de Dedicação (MD). Na matriz, cada célula indica o quanto um funcionário irá se dedicar para concluir cada tarefa. As linhas representam os funcionários e as colunas representam as tarefas. A Figura 2 ilustra uma matriz de dedicação com três funcionários e quatro tarefas, onde o valor máximo é 1,0, significando que o funcionário se dedica apenas àquela tarefa e o valor 0,0 significa que o funcionário não está realizando aquela tarefa.

Figura 2 – Matriz de dedicação de um projeto.

	Tarefa 1	Tarefa 2	Tarefa 3	Tarefa 4
Funcionário 1	1,0	0,0	0,0	0,0
Funcionário 2	0,2	0,6	0,1	0,1
Funcionário 3	0,0	0,0	0,3	0,7

Fonte: Próprio autor.

Segundo Alba e Chicano (2007) o principal objetivo do SPSP é associar funcionários a tarefas de maneira que atenda às restrições do problema e a duração e custo do projeto sejam minimizados.

Para solucionar o SPSP é necessário respeitar três restrições relacionadas aos funcionários e as tarefas, definidas a seguir:

Restrição 1 - Mínimo de funcionário em um tarefa: Cada tarefa deve ser realizada por pelo menos uma pessoa. Assim cada elemento da matriz de dedicação md_{ij} tem que ser maior que zero em todas as tarefas.

$$\sum_{i=1}^{n} md_{ij} > 0 \; \forall_j \in \{1,2,...,m\} \tag{2.1}$$

Restrição 2 - Habilidades do funcionário: O conjunto de habilidades exigido por uma tarefa deve ser incluído na união das habilidades dos funcionários que realizam a tarefa.

$$t_j^{habilidade} \subseteq \bigcup_{i=1} e_i^{habilidade} \; \forall_j \in \{1,2,...,m\} \tag{2.2}$$

Restrição 3 - Carga de trabalho: Nenhum funcionário deve exceder a sua máxima dedicação ao projeto, ou seja, todo planejamento deve ser feito sem considerar horas extras. Para verificar essa restrição, primeiro tem-se que calcular a função de trabalho de cada funcionário, em relação ao tempo de execução do projeto (τ):

$$e_i^{trabalho}(\tau) = \sum_{\{j|t_j^{inicio}<\tau<t_j^{f}im\}} m_{ij} \tag{2.3}$$

O funcionário excede o seu limite de trabalho caso $e_i^{trabalho}$ for maior que a dedicação máxima do funcionário em qualquer instante τ da execução do projeto. Para as soluções válidas, ou seja, que atendam às restrições do problema, são calculados o início e fim de cada tarefa, a duração e custo do projeto, como dado a seguir.

$$t_j^{inicio} = \begin{cases} 0 & \text{se } \forall k \neq j, (t_k, t_j) \notin A \\ max\{t_k^{fim} | (t_k, t_j) \in A & caso \text{ contrário} \end{cases} \tag{2.4}$$

$$t_j^{fim} = t_j^{inicio} + t_j^{dur} \tag{2.5}$$

O cálculo da duração, da tarefa e do projeto pelas fórmulas 2.6 e respectivamente 2.7.

$$t_j^{dur} = \frac{t_j^{esforco}}{\sum_{i=1}^{n} md_{ij}} \tag{2.6}$$

O cálculo da duração do projeto é dado por

$$P_{dur} = max\{t_j^{fim} \mid \forall\, k \neq j(t_j, t_k) \;\notin A\} \tag{2.7}$$

O custo de cada tarefa, é a multiplicação da soma dos salários de todos os funcionário da tarefa, pela dedicação de cada um md_{ij} multiplicado pelo duração da tarefa t_j^{dur}, dado pela fórmula 2.8, e o custo total do projeto P_{custo} é dado pelo somatório dos custo das tarefas mostrado na fórmula 2.9

$$t_j^{custo} = \sum_{i=1}^{n} e_i^{salario} \;.\; md_{ij}.t_j^{dur} \tag{2.8}$$

$$P_{custo} \sum_{i=1}^{m} t_j^{custo} \tag{2.9}$$

Como o SPSP é um problema classificado como NP-difícil (BIJU; MOHANASUNDARAM, 2015), encontrar uma solução ótima com métodos exatos implica em custo computacional muito grande. Uma alternativa para lidar com estes problemas é utilizar meta-heurísticas para se encontrar soluções ótimas ou quase ótimas, como discutido na próxima seção.

2.1.1 Algoritmos Heurísticos para o SPSP.

Para resolver o SPSP é necessário modelá-lo formalmente como um problema multiobjetivo. Um problema é considerado multiobjetivo se possui k objetivos a serem otimizados ao

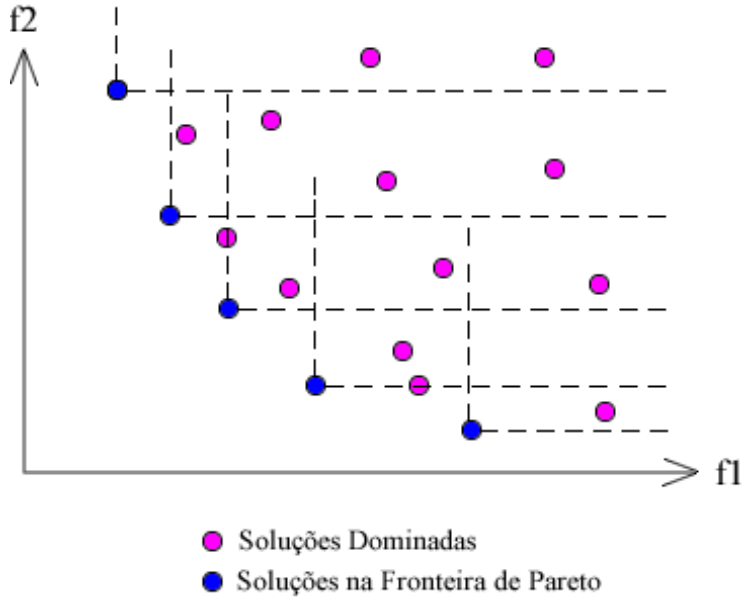
mesmo tempo, sendo $k \geq 2$. Matematicamente, uma otimização com vários objetivos é composta por vetores de decisões $x = [x_1, ..., x_s]^{\mathscr{T}}$, onde s são variáveis de decisões e $\mathscr{T}$ é a transposição do vetor coluna. Normalmente, problemas multiobjetivos possuem restrições *constraints*, representadas por inequações $g_i \leq 0, i = \{1, ..., l\}$ ou igualdades $h_j(x) = 0, j = \{1, ..., p\}$ onde l e p representam a quantidade de restrições. Uma função objetivo é utilizada para medir a qualidade de uma solução, expressa por $f(x) = [f_1(x), ..., f_z(x)]^{\mathscr{T}}$, no qual z é o número de objetivos (COELLO; LAMONT; VELDHUIZEN, 2007).

A Fronteira de Pareto é a representação no espaço de objetivos de um conjunto de soluções que possuem um bom compromisso entre todos os objetivos. (GOLDBARG; GOLDBARG; LUNA, 2017). Para Coello, Lamont e Veldhuizen (2007), define-se matematicamente:

Definição 2.1 ***(Dominância de Pareto)*** *Uma solução $u = (u_1, ..., u_z)$, domina uma solução $v = (v_1, ..., v_z)$ se e somente se* z ∈ *K = (1,..,k) leva a* $f_k(u) \ni f_k(v)$ *e* $\exists$ *z:*$f_k(u) < f_k(v)$.

.

Figura 3 – Gráfico com a fronteira de Pareto.



Fonte: COELLO; LAMONT; VELDHUIZEN (2006), adaptado.

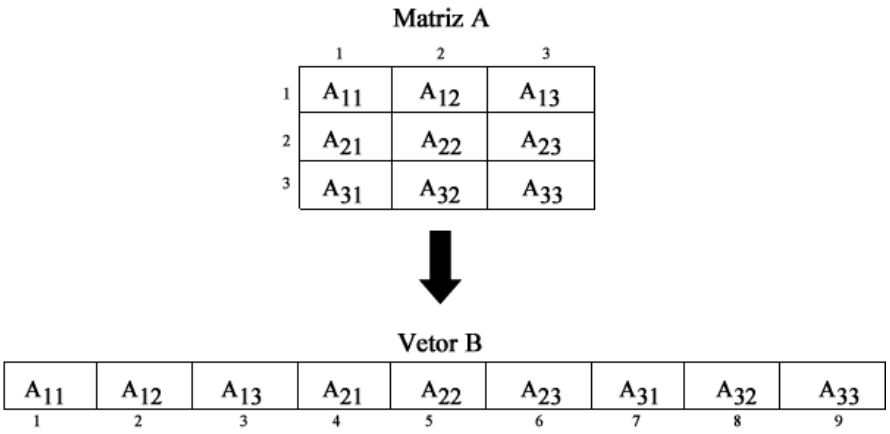
O gráfico da Figura 3 ilustra a relação de dominância entre soluções em um plano cartesiano, onde cada eixo representa um objetivo, sendo f_1 e f_2 objetivos a serem minimizados. As soluções em azul formam a Fronteira de Pareto, logo segundo a definição 2.1, não existe uma solução que faça parte da Fronteira de Pareto que domine outra por completo, ou seja, para qualquer solução da Fronteira de Pareto, existe ao menos uma outra solução que é melhor que ela em um objetivo. Apesar da avaliação de soluções multiobjetivos serem difíceis, é necessário

utilizar alguma métrica para definir qual solução utilizar, uma delas é o hipervolume descrito na seção 2.1.2.

Uma das formas de reduzir o custo computacional para solucionar o SPSP consiste em aplicar técnicas de busca meta-heurísticas. Entretanto, para utilizar estas técnicas para resolver problemas de Engenharia de Software é preciso formular o problema como um problema de busca e segundo Harman e Jones (2001) faz-se necessário definir três pontos principais: uma boa representação para o problema, um conjunto de operadores que permitam manipular as soluções e uma função aptidão que calcule quão boa é a solução (*fitness*).

Como mencionado na seção 2.1, no SPSP as soluções são representadas por matrizes de dedicação, convertidas para vetores unidimensionais para permitir o uso dos operadores evolutivos utilizados pelos algoritmos utilizados neste trabalho. A Figura 4 ilustra a conversão de uma matriz A de tamanho n x m em um vetor B, começando de cima para baixo, colocando o primeiro item da segunda linha i após o último elemento da linha i-1, $2 < i \leq$ n. Por exemplo, a posição A_{23} da matriz foi mapeada para a posição B_6 do vetor.

Figura 4 – Transformação de matriz em vetor.



Fonte: Próprio autor (2018)

Os operadores de manipulação dos algoritmos variam de acordo com qual algoritmo está sendo utilizado. Na literatura existem diversos algoritmos multiobjetivo que podem ser utilizados para resolver o SPSP, dentre eles podemos destacar o SMPSO (KENNEDY; EBERHART, 1995), o PAES (KNOWLES; CORNE, 1999) e o NSGA-II (DEB et al., 2002) que são utilizados neste trabalho.

O SMPSO (*Speed constrained Multi-objective PSO*) é a versão multiobjetivo do algoritmo Enxame de Partículas (PSO). Funciona de forma a simular o comportamento social de um bando de pássaros adicionando um operador de turbulência e através da comunicação de seus membros, visando determinar qual a melhor direção a ser tomada na busca de uma solução.

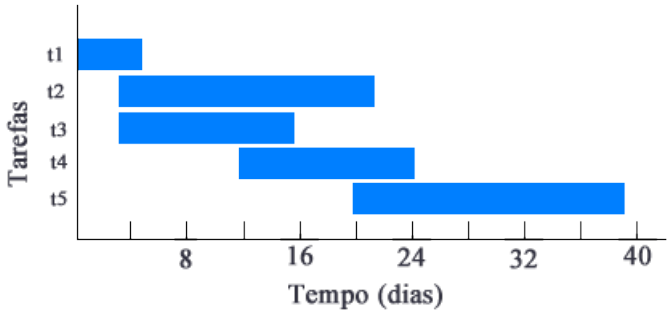
O NSGA-II (*Nondominated Sorting Genetic Algorithm II*) é um algoritmo genético baseado na geração de uma nova população a partir da população original, aplicando os operadores genéticos típicos (seleção, *crossover* e mutação). Os indivíduos da nova e velha população são selecionadas de acordo sua classificação, e as melhores soluções são escolhidas para criar uma nova população.

O PAES (*Pareto Archived Evolution Strategy*) é um algoritmo baseado em uma estratégia de evolução simples *(1 + 1)*, é capaz de encontrar diversas soluções no Conjunto Ótimo de Pareto. A partir de uma única solução, o PAES é capaz de se aproximar toda a Fronteira de Pareto, para isso ele utiliza um operador de mutação que gera soluções nova não dominadas.

A aptidão de uma solução é o mecanismo que guia o algoritmo no espaço de busca. Ela é calculada com base nas métricas referentes ao problema, no SPSP uma boa solução é aquela que seguindo as restrições, minimiza o tempo e custo do projeto.

Uma das formas de visualizar uma solução do SPSP, é utilizando o gráfico de Gantt, que permite ilustrar a disposição das tarefas em relação ao tempo, mostrando o início, fim e duração de uma tarefa, utilizando barras sobre o eixo horizontal do gráfico. A Figura 5 mostra um exemplo de gráfico de Gantt, com cinco tarefas ao longo de 39 dias.

Figura 5 – Gráfico de Gantt



Fonte: Próprio autor.

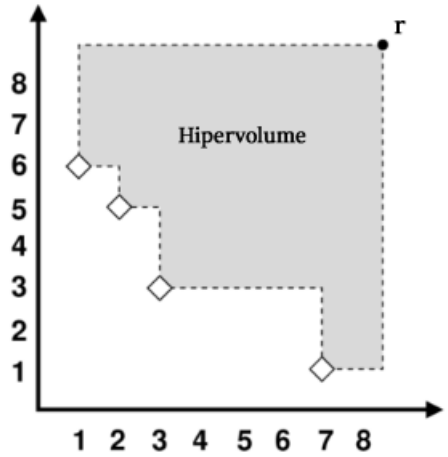
2.1.2 Hipervolume.

Avaliar a qualidade de soluções em um problema multiobjetivo não é uma tarefa simples. Por isso, faz-se necessário ter uma métrica que estabeleça quais soluções devem ser mantidas e quais serão excluídas do conjunto de soluções.

A métrica utilizada neste trabalho foi o hipervolume (BEUME et al., 2009). O hipervolume calcula a área entre a fronteira de Pareto usando um ponto de referência *r*, que é um ponto não alcançado por nenhuma das soluções. A área resultante tem um formato de um polígono e quanto maior esse polígono. melhor. A área em cinza da Figura 6 mostra graficamente o hipervolume, os quadrados são as soluções que compõem a Fronteira de Pareto e *r* o ponto de

referência. O ponto *r* é um valor não alcançado por nenhuma solução, por exemplo utilizando dados normalizados, em um intervalo de 0 à 1, um bom valor para *r* é 1.1.

Figura 6 – Representação do cálculo do hipervolume.



Fonte: Adaptado de (Lopez-ibanez).

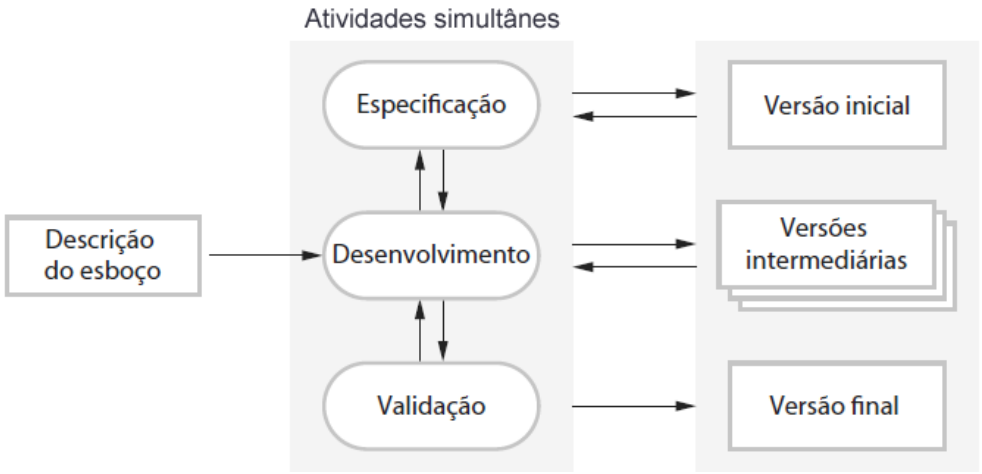
2.2 Metodologia de Desenvolvimento de Software Utilizada no Projeto.

Este projeto utiliza uma metodologia de desenvolvimento incremental. O desenvolvimento incremental é baseado na ideia de que o projeto é desenvolvido em várias iterações. Primeiramente, cria-se uma versão inicial, em seguida expõe-se esta versão à crítica dos usuários e a partir dos resultados da avaliação preliminar, continua-se o desenvolvimento, repetindo-se este processo e criando-se várias versões até que a aplicação tenha um resultado satisfatório (SOMMERVILLE, 2011). Dessa forma, as atividades do projeto são intercaladas; a especificação, o desenvolvimento e validação ocorrem durante todo o tempo de vida do projeto. Neste trabalho a atividade de especificação da aplicação é feita utilizando-se a linguagem de modelagem UML brevemente descrita na seção 2.3.

A Figura 7 ilustra esse método de desenvolvimento, no qual o desenvolvimento começa a partir de um esboço e segue alternando entre as atividades, chegando-se a uma versão final.

O desenvolvimento incremental possui pontos positivos e negativos, como qualquer outro método. Segundo Sommerville (2011) os pontos positivos do modelo incremental são: o custo reduzido para mudanças nos requisitos de um sistema, a facilidade para descobrir os interesses do cliente e a possibilidade de entregar parte do projeto para que o cliente possa usufruir antes mesmo do projeto ter terminado. Como pontos negativos, pode-se destacar a dificuldade de documentar as várias fases do desenvolvimento do sistema e principalmente os danos causados

Figura 7 – Diagrama de funcionamento do desenvolvimento incremental.



Fonte: Sommerville (2011).

ao sistema pela adição continua de novas funcionalidades, fazendo com que ele se degrade com o tempo, a menos que se gaste tempo e dinheiro para fazer a refatoração do sistema.

2.3 A Linguagem UML.

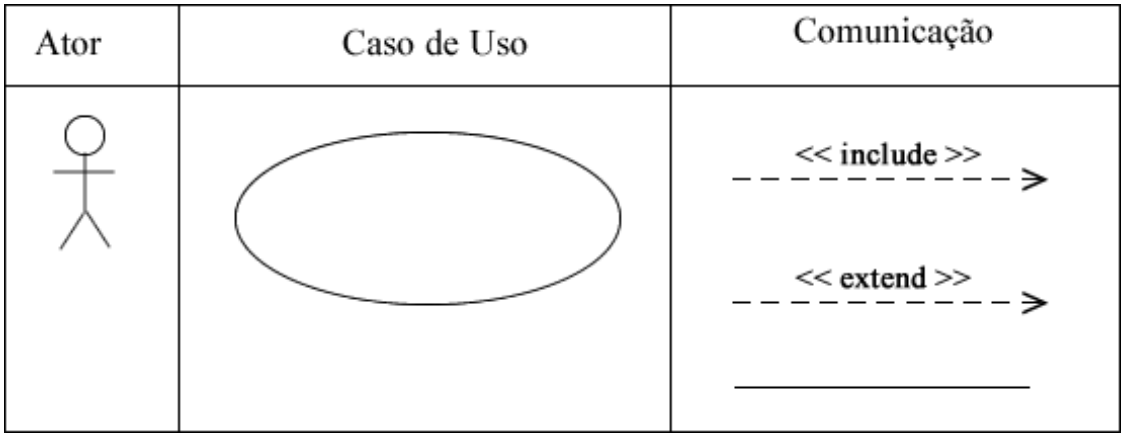
Para confeccionar todos os diagramas necessários para essa aplicação foi utilizado a Linguagem de Modelagem Unificada (em inglês, *Unified Modeling Language* (UML)). Define-se UML como uma família de notações gráficas, apoiada por um modelo único, que ajuda na descrição e no projeto de sistemas de software, principalmente os que utilizam orientação a objetos (FOWLER; SCOTT, 2000). Nesta seção introduziremos a notação UML estritamente necessária para o entendimento da monografia.

Para expressar os requisitos deste trabalho foi feito o diagrama de casos de uso (JACOBSON et al., 1993). Os diagramas de casos de uso são compostos de atores, caso de uso e comunicação/relacionamento entre os elementos do diagrama. Os atores são pessoas ou dispositivos que utilizam o sistema (são usuários do sistema). Caso de uso é uma tarefa ou uma funcionalidade realizada por um ator. Comunicação é a relação entre atores e a conexão entre atores e casos de uso. Cenário é uma sequência de eventos acionados durante a interação dos usuários com o sistema. Segundo Pressman (2011) um caso de uso conta uma história sob o ponto de vista do usuário final, indicando como ele interage com o software ou sistema.

Os relacionamentos em um diagrama de casos de uso podem ser: associação, que demonstra que o ator utiliza o caso de uso ligado a ele; generalização, que expressa a herança de casos de uso entre atores; *include* que é um relacionamento entre casos de uso, de forma que se A faz um *include* para B, então B é essencial para A e *extend* que indica que se B *extend* A, então B pode ser acrescentado ao comportamento de A.

A representação gráfica no UML usado neste trabalho pode ser vista na Figura 8,onde o ator é representado por um boneco, os casos de uso são representados por elipses, a relação de associação é uma linha simples, o *include* e *extend* são representados por setas pontilhadas com o seu respectivo « nome » sobre elas.

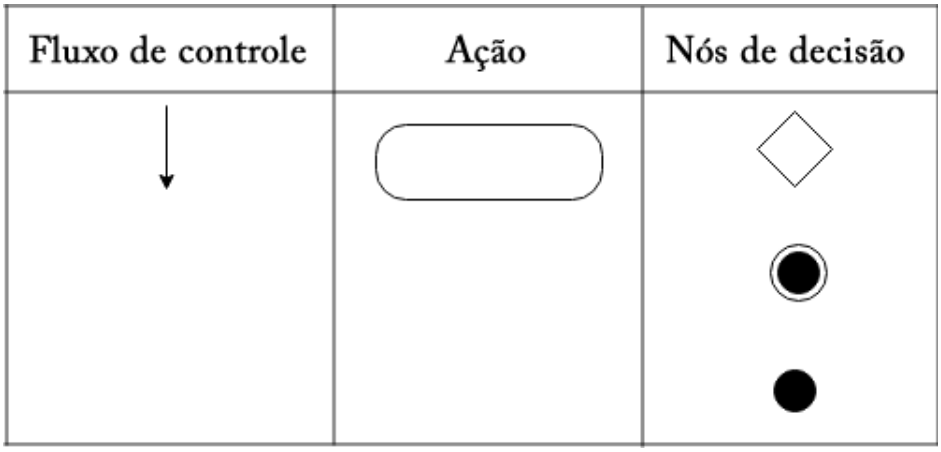
Figura 8 – Notação UML - diagrama casos de uso.



Fonte: Próprio autor (2018).

Além do o diagrama de casos de uso, neste trabalho utilizamos os diagramas de atividades e classes. O diagrama de atividades em UML permite ao leitor ver o fluxo de interação em um cenário específico. O diagrama de atividades é similar a um fluxograma. O diagrama utiliza retângulos com cantos arredondados para representar uma função específica do sistema (ação), setas para indicar o fluxo que o usuário utiliza no sistema, losangos para indicar pontos em que o usuário deve escolher qual fluxo o programa seguirá, um círculo preenchido, indicando o início do diagrama e um círculo preenchido dentro de um círculo vazio indicando o fim do diagrama (PRESSMAN, 2011). A Figura 9 mostra os símbolos utilizados no diagrama de atividades.

Figura 9 – Notação UML - diagrama de atividades

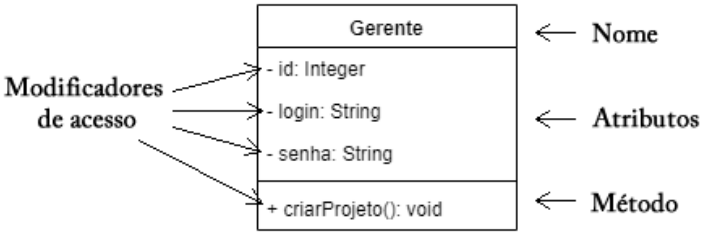


Fonte: Próprio autor (2018).

Os diagramas de classes são utilizados para desenvolver sistemas orientados a objetos, com o objetivo de mostrar as classes do sistema e a associação entre elas. Uma associação é uma conexão entre classes indicando algum relacionamento entre as classes. (SOMMERVILLE, 2011)

As classes do diagrama são representadas um por um retângulo com três divisões, a superior indicando o nome, a do centro indicando os atributos das classes, e a divisão inferior indicando os métodos que pertencem à classe. Os atributos e métodos possuem um nível de visibilidade, podendo ser, privado (-) , protegido (#) ou público (+). Os atributos privados só podem ser acessados de forma direta pela própria classe, os atributos protegidos são acessados pelas classes do mesmo pacote e os atributos públicos acessados por qualquer classe. A Figura 10 exemplifica o formato e os componentes da classe no diagrama.

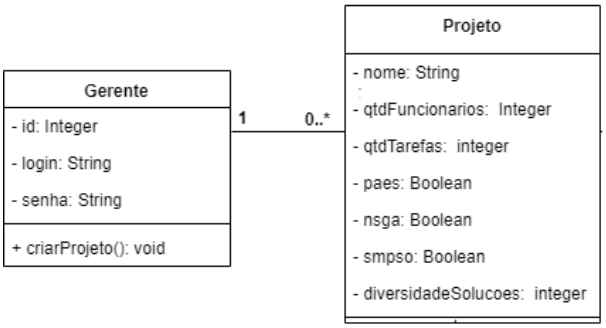
Figura 10 – Diagrama de classes - componentes de uma classe.



Fonte: Próprio autor.

No diagrama, as classes podem se relacionar formando associações, onde uma classe faz referência outra, utilizando uma linha sólida para representá-la graficamente. Em cada ponta da linha utiliza-se uma notação indicando o intervalo de objetos que participam dessa relação, que pode tem uma faixa de variação de zero a um valor indeterminado representado por (*), ou ser um valor exato. Por exemplo, na Figura 11 um objeto da classe gerente está associada a 0 ou mais objetos da classe projeto.

Figura 11 – Diagrama de classes - multiciplidade.



Fonte: Próprio autor.

2.4 A Arquitetura MVC

Para desenvolver este trabalho foi utilizado o padrão de arquitetura de software Model-View-Controller, conhecido em português por Modelo-Visão-Controlador (MVC), proposto por Reenskaug (1979), com o objetivo de criar uma ponte entre a maneira que usuário pensa e a maneira que o computador funciona, separando a aplicação em três componentes, permitindo assim que os usuários controlem de forma mais organizada um conjunto grande e complexo de dados.

A camada lógica referente ao modelo, é a principal camada de uma aplicação, ela modela o problema desejado e possui lógica para resolvê-lo. É uma camada de alto nível, que abstrai as estruturas do banco de dados, de maneira a deixar que os banco de dados armazenem e garanta a integridade dos dados de forma independente (KUPP; MAKRIS, 2012). Esta camada é responsável por instruções de leitura, escrita e validação dos dados

A camada apresentação ou visualização, contém todas as funções específicas à interface e possibilita a apresentação do conteúdo e lógica de processamento, inclusive todos os objetos de conteúdo, acesso a fontes de dados/informações externas e toda a funcionalidade de processamento requerida pelo usuário final (PRESSMAN, 2011).

A camada de controle da arquitetura MVC combina e gerencia o acesso aos modelos e entradas do usuário (por meio da camada de apresentação), e usa essas entradas para executar ações nos dados encapsulados nos modelos (KUPP; MAKRIS, 2012). Juntas, essas três camadas definem de forma clara onde está cada componente da aplicação, de maneira a criar no usuário a ilusão de que a memória do computador é uma extensão de sua memória.

Um exemplo de utilização da arquitetura MVC nesta aplicação é o cadastro das informações do projeto que são interpretadas pela camada de controle. A camada de visualização é formada por telas como, exibição da matriz de dedicação e o mapa de calor com alocação de recursos, que representam os mesmos dados de uma solução de forma diferentes. Os dados cadastrados e as soluções geradas são gerenciados pela camada lógica.

2.5 Ferramentas e linguagens utilizadas no desenvolvimento do projeto.

O serviço *online* do Google draw.io[1] foi utilizado para confeccionar o diagrama de casos de uso, diagrama de sequência e o diagrama de classes seguindo os padrões da linguagem UML, descrita na seção 2.3 para representar a aplicação graficamente.

Para desenvolver a aplicação foi utilizado a linguagem de programação Java[2], pois é uma

[1] Disponível em: <https://www.draw.io/>.
[2] Disponível em: <https://www.oracle.com/technetwork/java/index.html>.

linguagem orientada a objetos, independente de plataforma e principalmente devido ao fato de ser a linguagem que o *framework* jMetal foi desenvolvido.

O jMetal[3] é um *framework* orientado a objetos que visa desenvolver estudos e experimentos de problemas de otimização multiobjetiva utilizando algoritmos metaheurísticos. O jMetal possui um conjunto de meta-heurísticas amplamente utilizadas na literatura, um conjunto de problemas computacionalmente difíceis e um conjunto de indicadores de qualidade para avaliar o desempenho dos algoritmos (DURILLO; NEBRO, 2011).

Em conjunto com o Java, foi utilizado o Swing[4], uma API (em inglês, *Application Programming Interface*) que facilita o desenvolvimento de interface gráficas. O Ambiente de Desenvolvimento Integrado (em inglês, *Integrated Development Environment* (IDE)) Eclipse[5] foi utilizado para desenvolver a aplicação.

2.6 Trabalhos Relacionados

A fim de entender os problemas em aplicação similares à ferramenta proposta, foram analisadas diversas iniciativas de projetos que envolvem o planejamento de um projeto de software, a seguir brevemente descritas.

O Microsoft Project [6] é uma ferramenta profissional para gerenciamento de projetos, que auxilia os gerentes de projeto, através de funcionalidade. Conhecido como MS Project, ele possui diversas funcionalidades, entre elas podemos destacar, gerenciamento de calendários, atribuição de custos de recursos e monitoramento e controle da execução de tarefas do projeto.

O Primavera P6 [7]é um pacote de software profissional para planejamento e controle de múltiplos projetos. Possui as funcionalidade, o CPM para realizar o agendamento de projetos e distribuição de recursos.

O RESCON (RESource CONstrained) (DEBLAERE; DEMEULEMEESTER; HERROELEN, 2011) é uma ferramenta gratuita e com cunho educativo utilizada pela Universidade Católica de Leuven, na Bélgica. Seu principal foco é o Problema do Escalonamento e Atribuição de Tarefas em Projetos de Software com Restrição de Recursos (em inglês, *Resource-Constrained Project Scheduling Problem* (RCPSP)). Fornece algumas funções úteis para o planejamento do projeto como: a visualização do projeto como diagrama de rede, recursos utilizados em uma tarefa e o diagrama de Gantt para ver a distribuição e precedência de tarefas. Para fazer o planejamento o RESCON permite a utilização de algoritmos clássicos como *branch and bound* e busca tabu, além de permitir que o usuário utilize seus próprios algoritmos.

[3] Disponível em: <http://jmetal.sourceforge.net/>.
[4] Disponível em: <https://docs.oracle.com/javase/8/docs/technotes/guides/swing/index.html>.
[5] Disponível em: <http://www.eclipse.org/>.
[6] Disponível em: <https://products.office.com/en/project/project-and-portfolio-management-software>.
[7] Disponível em: <https://www.oracle.com/applications/primavera/products/project-portfolio-management/>.

O PpcProject (SALAS-MORERA et al., 2013) é uma ferramenta gratuita e de código aberto para uso educacional no gerenciamento de projetos de software. Possibilita a criação de diagrama de rede do projeto, a criação de gráficos Gantt e gráficos PERT, a identificação de atividades críticas e caminhos críticos .

IntelliSPM (STYLIANOU; GERASIMOU; ANDREOU, 2012) é uma ferramenta que visa prover apoio às atividades de gerenciamento de projetos. A ferramenta compreende vários mecanismos de otimização utilizados na área de inteligência computacional. O objetivo do IntelliSPM é oferecer sugestões para gerentes de projeto contendo um conjunto de possíveis cronogramas de projetos e estratégias de pessoal que minimizam duração e maximizam o uso de recursos . Dentre os recursos do IntelliSPM estão, por exemplo, exibição da tabela com todas as soluções e os valores de sua função objetivo e restrição, plotagem da fronteira de Pareto, matriz de alocação de recursos, gráfico de uso de recursos.

A IA-GPS, se assemelha a todas as ferramentas listadas, nas suas funcionalidades básicas, como o gráfico de Gantt, o mapa de alocação de recursos, grafo de dependências. Assemelha-se mais ao IntelliSPM, já que ambos utilizam meta-heurísticas para gerar o escalonamento das tarefas. Entretanto a IA-GPS se diferencia pela possibilidade de importar projetos de outras ferramentas e exportar soluções, e por possuir um sistema de controle de usuário, além de prover integração com o *framework* jMetal.

A tabela 3 sumariza as principais características pertencentes ao IA-GPS , confrontando-as com os trabalhos relacionados.

Tabela 3 – Características das ferramentas.

Funcionalidades	IA-GPS	MS Project	Primavera P6	RESCON	PpcProject	IntelliSPM
Importar projeto	✓	✓	✓	X	X	X
Exportação de soluções	✓	✓	✓	X	X	X
Gráfico de Gantt	✓	✓	✓	✓	✓	✓
Gráfico de dependências	✓	✓	✓	✓	✓	✓
Mapa de alocação de recursos	✓	✓	✓	✓	X	✓
Matriz de dedicação	✓	X	X	X	X	X
Meta-heurística	✓	X	X	X	X	✓
Integração com JMetal	✓	X	X	X	X	X

Fonte: Próprio autor.

3

A Ferramenta IA-GPS

Este capítulo descreve a elaboração da ferramenta IA-GPS (Interface de Apoio a Gerentes de Projetos de Software) de acordo com a metodologia descrita na seção 1.1.2 e utilizando a notação introduzida na seção 2.3.

3.1 Levantamento de Requisitos

Define-se engenharia de requisitos como o processo de descobrir, analisar, documentar e verificar funções e restrições. O conjunto de requisitos de um sistema descreve o que o sistema deve fazer, os serviços que oferecem e as restrições a seu funcionamento, além de refletirem as necessidades dos clientes de um determinado sistema. Normalmente os requisitos de software são classificados de duas formas, como requisitos funcionais e requisitos não funcionais (SOMMERVILLE, 2011).

3.1.1 Requisitos funcionais e não funcionais

Segundo Sommerville (2011) os requisitos funcionais são declarações de serviços de como o sistema deve reagir e se comportar de acordo com situações específicas. Neste trabalho foi elaborado dez requisitos funcionais, tentando preencher ao máximo as necessidades do gerente de projeto de software, de maneira que ele possa ter em *mãos* as ferramentas adequadas para auxiliar na elaboração de um bom planejamento. A Tabela 4 mostra estes requisitos.

Tabela 4 – Requisitos funcionais da aplicação

Código	Requisito
RF01	Cadastrar projeto
RF02	Incluir/Excluir/Alterar dados do projeto
RF03	Escolher parâmetros do algoritmo
RF04	Escolher meta-heurísticas
RF05	Visualizar gráficos com as Fronteiras de Pareto
RF06	Visualizar o Diagrama de Gantt para cada solução
RF07	Visualizar o grafo de dependências entre as tarefas
RF08	Visualizar gráfico de alocação de recursos (tarefa/programador)
RF09	Visualizar alocação de recursos da tarefa

Fonte: Próprio autor.

Os requisitos não funcionais são restrições aos serviços ou funções oferecidas pelo sistema. Incluem restrições de tempo, restrições no processo de desenvolvimento e restrições impostas pelas normas ou leis e normalmente aplicam-se ao sistema como um todo (SOMMERVILLE, 2011). Para que a aplicação funcione de acordo com as restrições de desenvolvimento, foram definidos cinco requisitos não funcionais, que podem ser vistos na Tabela 5.

Tabela 5 – Requisitos não funcionais da aplicação

Código	Requisito
RNF01	O programa deve poder ser integrado ao jMetal
RNF02	O programa deve funcionar com várias meta-heurísticas
RNF03	O programa deve guardar os resultados em um banco de dados
RNF04	O programa deve poder exportar as soluções em formato xml e xls
RNF05	O programa deve poder importar informações de outros projetos em formato xls

Fonte: Próprio autor.

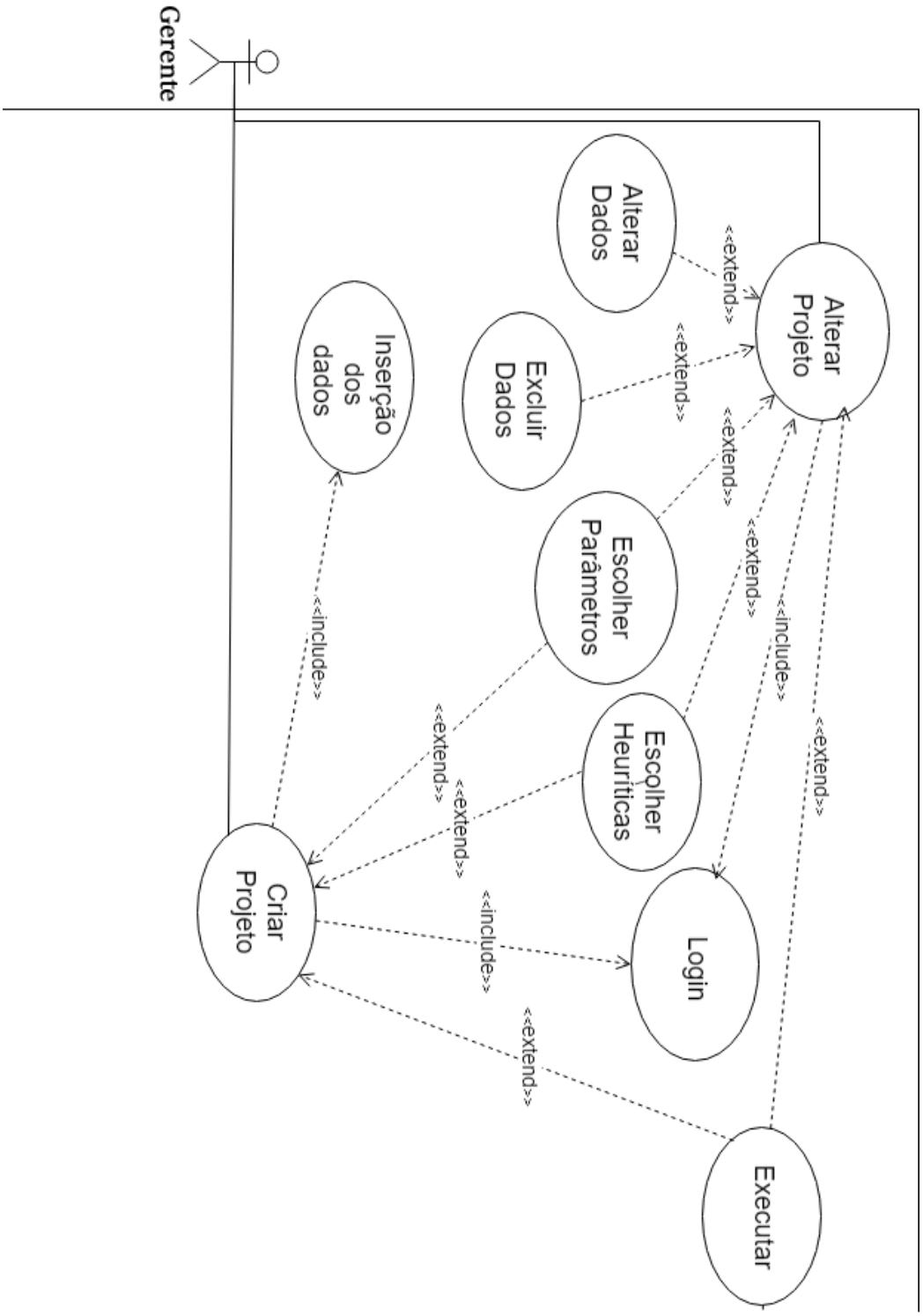
3.1.2 Diagrama de Casos de Uso

Neste trabalho, o diagrama de caso de uso formado pelas Figuras 12 e 13 finalizam a fase de levantamento de requisitos. Foi identificado apenas um ator, o gerente de projeto de software. Ele pode criar ou alterar um projeto, após realizar sua autenticação no sistema. A partir disso, ele pode escolher as meta-heurísticas e seus parâmetros, além de alterar ou adicionar novos dados. Caso ele não escolha nenhuma meta-heurística e/ou seus parâmetros a aplicação irá funcionar no modo pré-configurado (*default*).

Em seguida, o gerente de projeto pode executar a aplicação, e após os algoritmos produzirem os resultados, o gerente pode visualizá-los e analisá-los. A ferramenta possibilita ao

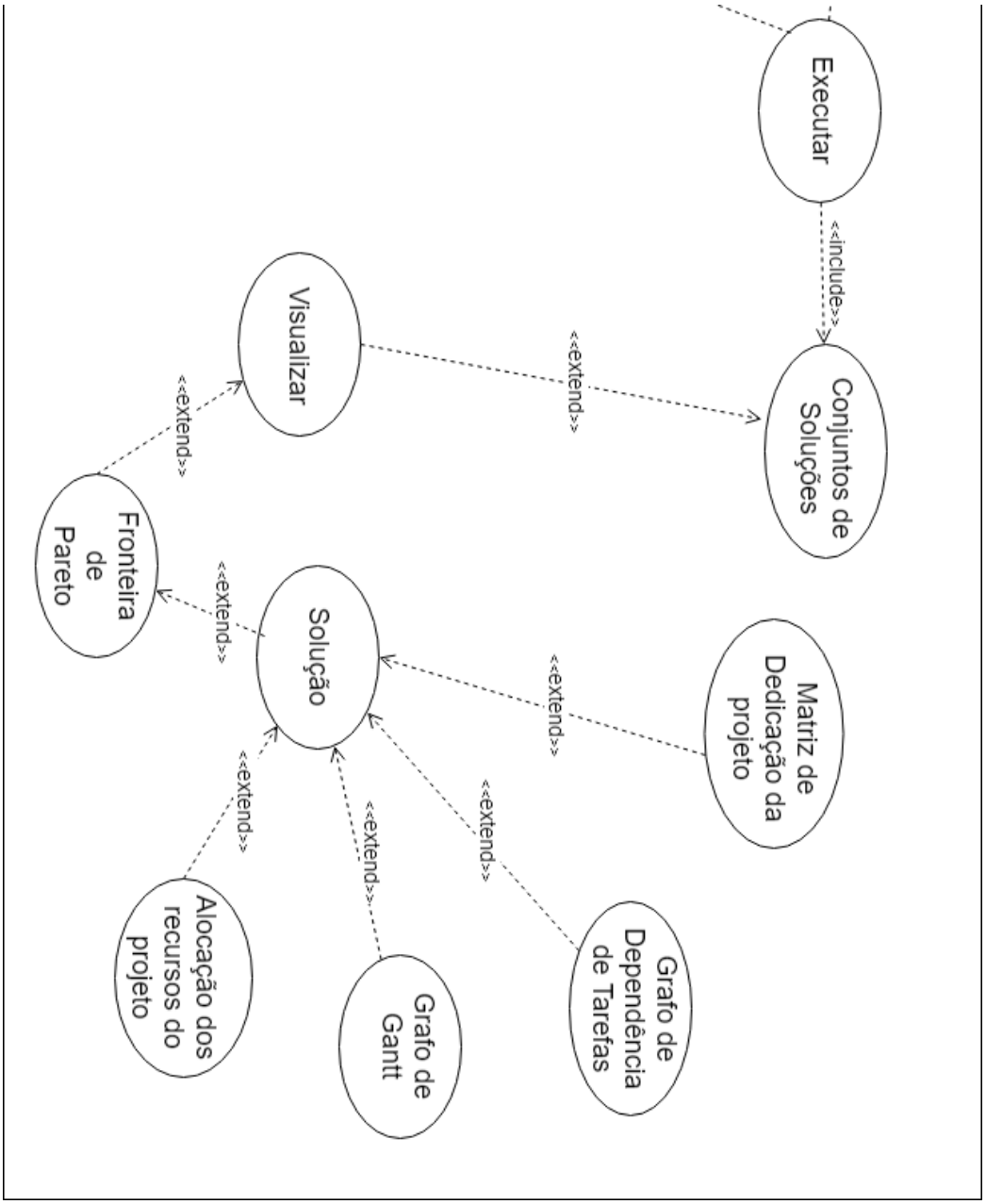
gerente ver o conjunto de soluções e a distribuição das soluções na fronteira de Pareto e analisar as soluções de forma mais detalhada através de gráficos como, gráfico de Gantt, gráficos de alocação de recursos da tarefa e projeto como um todo. Além disso, permite a visualização da matriz dedicação de um funcionário e o grafo de dependências entre as tarefas.

Figura 12 – Diagrama de casos de uso da aplicação - parte 1.



Fonte: Próprio autor.

Figura 13 – Diagrama de casos de uso da aplicação - parte 2.



Fonte: Próprio autor.

Para melhorar o entendimento do funcionamento e melhorar a documentação do projeto da IA-GPS, se faz necessário descrever os casos de uso de maneira a detalhar seu funcionamento. Portanto, para caso de uso do diagrama das Figuras 12 e 13 foi feito uma descrição detalhada de sua estrutura. Para ilustrar esta atividade, a Figura 14 mostra os detalhes do caso de uso Cadastrar Projeto.

Figura 14 – Descrição de caso de uso - Cadastrar Projeto

Caso de Uso (CSU3):	Cadastrar Projeto.
Objetivo:	Utilizar os dados do projeto para prover escalonamentos de tarefas e funcionários.
Requisito Funcional:	RF01.
Prioridade:	Alta.
Frequência de uso:	Alta.
Ator Principal:	Gerente.
Pré-Condições:	Ter feito login na ferramenta.
Pós-Condições:	Os dados do projeto são armazenados no sistema.
Fluxo Principal:	1. O gerente escolhe a opção novo projeto na tela de usuário 2. O sistema mostra uma janela de cadastro. 3. O gerente cadastra as habilidades necessárias para a conclusão do projeto. 4. O gerente cadastra os funcionários que estão disponíveis para executar o projeto, especificando os seus nomes, habilidades e salário. 5. O gerente cadastra as tarefas que compõem o projeto, especificando seus nomes, custo, habilidades necessárias para realizá-la e as tarefas que necessitam serem realizadas antes dela. 6. O gerente escolhe quais algoritmos devem ser utilizados para gerar o escalonamento do projeto, a quantidade das soluções e sua diversidade. 7. O gerente confirma as suas escolhas. 8. O sistema informa se há alguma pendência no projeto.
Fluxo alternativo:	1. O gerente escolhe opção de importar projeto. 2. O sistema exibe uma janela para o gerente escolher o local de origem do projeto a ser importado. 3. O gerente escolhe o arquivo xml que contém o projeto. 4. O sistema importa as informações do projeto. 5. O gerente complementa manualmente as informações que faltarem no projeto.
Fluxo Exceção:	E1 - Gerente tentar cadastrar funcionario, tarefa ou habilidade existente. 1. Mensagem de erro, cancelar cadastro E2 - Gerente não cadastra funcionários com habilidades adequadas para realizar todas as tarefas. 1. Retorna para tela de cadastro de funcionários.

Fonte: Próprio autor.

3.2 Projeto da ferramenta

A IA-GPS foi proposta com objetivo de permitir que os gerentes de projetos de software utilizem algoritmos heurísticos de inteligência artificial, para realizar o escalonamento das tarefas e dos funcionários de um projeto de software de forma automatizada.

Com base no levantamento dos requisitos da seção 3.1, foi verificado a necessidade do gerente e foi projetado o diagrama de atividades 3.2.1 e o diagrama de classes 3.3 da IA-GPS.

3.2.1 Diagrama de Atividades

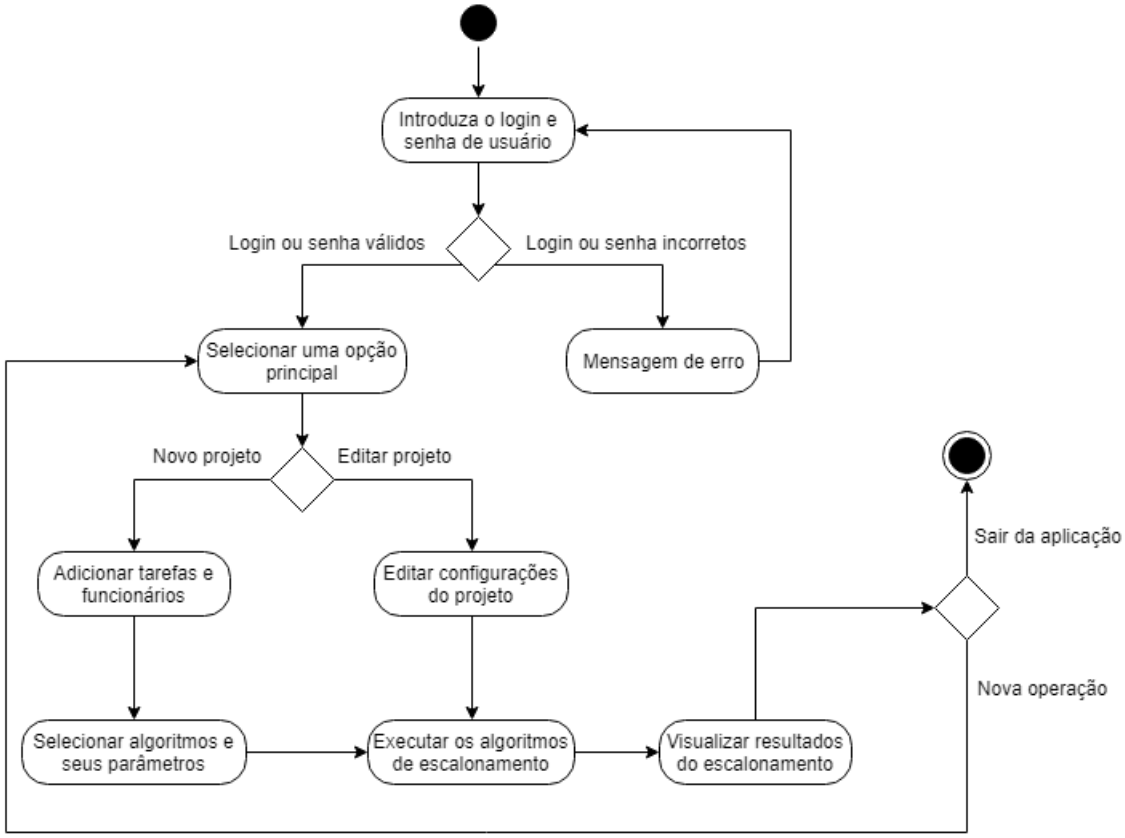
Para visualizar o fluxo da aplicação proposta por este trabalho, foi feito o diagrama de atividades, ilustrado na Figura 15. A aplicação possui um fluxo simples, com dois caminhos principais, um a partir da criação de um novo projeto e outro a partir da alteração de um projeto existente.

Ao criar um projeto o usuário necessita prover para a aplicação os dados básicos do projeto, como funcionários e tarefas. Após o preenchimento dos dados, o usuário pode optar por utilizar a configuração padrão da aplicação ou alterar, conforme sua necessidade, qual meta-heurística será utilizada, ou quais os parâmetros da meta-heurística que serão modificados. Em seguida, o usuário pode executar a aplicação. Quando o algoritmo finalizar a sua execução, são habilitadas as opções de visualização do resultado.

Através da aplicação, o usuário pode escolher várias maneiras de visualizar o resultado do escalonamento do projeto. Assim, é possível ver a Fronteira de Pareto com todas as soluções geradas pelo algoritmo, e selecionando-se uma solução, é possível ver o gráfico de Gantt desta solução, bem como o gráfico de alocação de seus recursos, o grafo de dependência das tarefas e a matriz de dedicação do projeto.

O segundo fluxo, como dito anteriormente, é similar ao primeiro, contudo, ao invés de inserir os dados, o gerente pode alterar as informações de um dado projeto. O restante dos passos são iguais.

Figura 15 – Diagrama de atividades



Fonte: Próprio autor.

3.3 Diagrama de Classes

O diagrama de classes, como mencionado na 2.3, ilustra os objetos do sistema, seus relacionamentos, métodos e atributos. O diagrama apresentado na Figura 16 mostra a organização principal do projeto; classes auxiliares foram omitidas para melhor entendimento geral da aplicação.

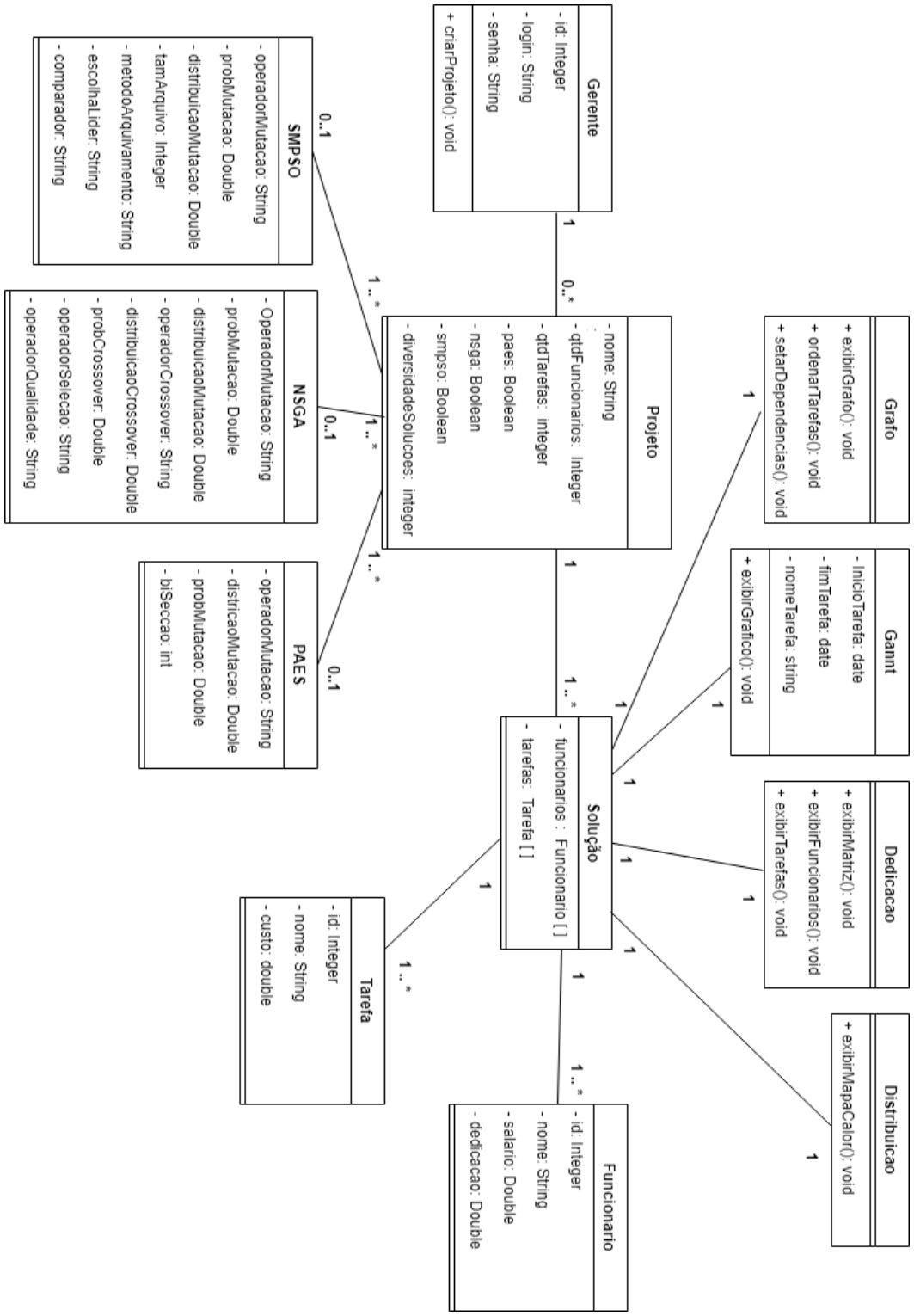
O diagrama mostra que o gerente de projeto possui atributos referentes ao seu *login* e a possibilidade de criar projetos. A classe de Projeto possui as características básicas do projeto, sendo responsável pelos parâmetros do problema, a quantidade de funcionários e tarefas, os parâmetros básicos referentes à utilização dos algoritmos, como quantas soluções devem ser geradas, o quão diferentes soluções podem ser durante a execução dos algoritmos e quais algoritmos serão utilizados para gerar os escalonamentos. Cada algoritmo possui uma classe própria por possuírem atributos distintos.

Um projeto pode ter diversas soluções e as soluções são formadas por pelos menos um funcionário e uma tarefa. Os funcionários têm como atributos id, nome, salário e dedicação. Já as tarefas possuem um id, um nome e um custo associado a ela.

As soluções podem ser visualizadas através das classes Grafo, Gantt, Dedicação e

Distribuição, cada uma com uma visão diferente da solução, com o objetivo de poder analisar qual solução é mais adequada para o gerente.

Figura 16 – Fragmento do diagrama de classes do projeto lógico



Fonte: Próprio autor.

3.4 Integração com o jMetal

O jMetal é um *framework* de otimização multiobjetivo baseado em meta-heurísticas. O framework foi escolhido para este trabalho por ser largamente utilizado na comunidade de SBSE, ser orientado a objetos, facilitando a extensibilidade e por ser bem documentado e fácil de ser integrado à ferramenta aqui desenvolvida.

Para utilizar os algoritmos SMPSO, PAES e NSGA-II, implementados no jMetal, foi necessário recuperar os dados da interface e criar uma maneira de atribuir os dados os algoritmos, através de classes de leitura e armazenamento do dados. Os resultados foram recuperados na classe principal de cada algoritmo. O mesmo processo ocorreu com o problema abordado, o SPSP, já que além de algoritmos, o jMetal também incorpora a implementação dos modelos dos problemas abordados na literatura.

A execução de cada algoritmo se dá apenas passando os parâmetros do problema, número de funcionários e de tarefas, e executando a chamada do método principal do algoritmo. As soluções, matrizes de dedicação, são recuperadas na classe do problema, e utilizadas nas classes da interface para exibir a soluções de forma prática ao gerente do projeto.

3.5 Principais funcionalidades da ferramenta

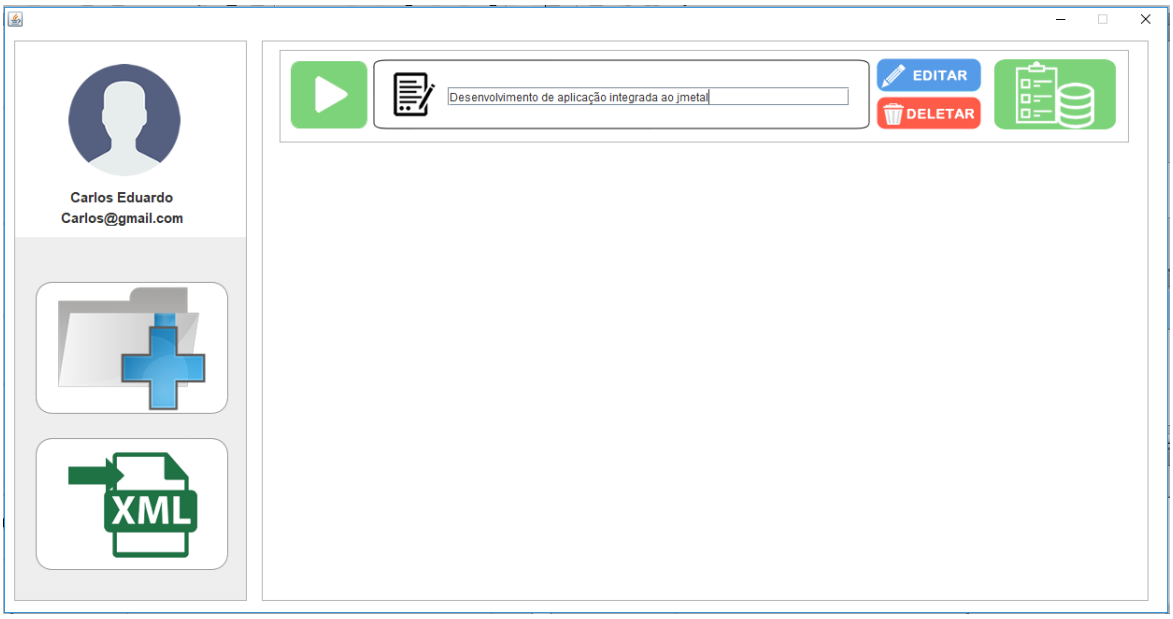
A ferramenta proposta tem o objetivo de facilitar o trabalho do gerente de projetos de software, e para isso foi implementado uma interface que permite utilizar os algoritmos SMPSO, NSGA-II e PAES para fazer os escalonamentos das tarefas de um projeto, durante a fase de planejamento.

Para analisar as soluções geradas, foram adicionadas funcionalidades que auxiliam o gerente do projeto escolher a solução mais adequada para o cenário. Nesta seção apresentaremos as principais telas e funcionalidades da ferramenta.

Ao logar na aplicação o gerente tem acesso a uma tela pessoal, com seus projetos, além de botões que permitem criar um novo projeto e importar dados xml de um projeto. É possível editar, excluir e executar um projeto conforme a vontade do gerente. Caso o projeto já tenha sido executado anteriormente, o gerente pode visualizar as soluções obtidas até aquele momento. A Figura 17 mostra a tela pessoal do gerente.

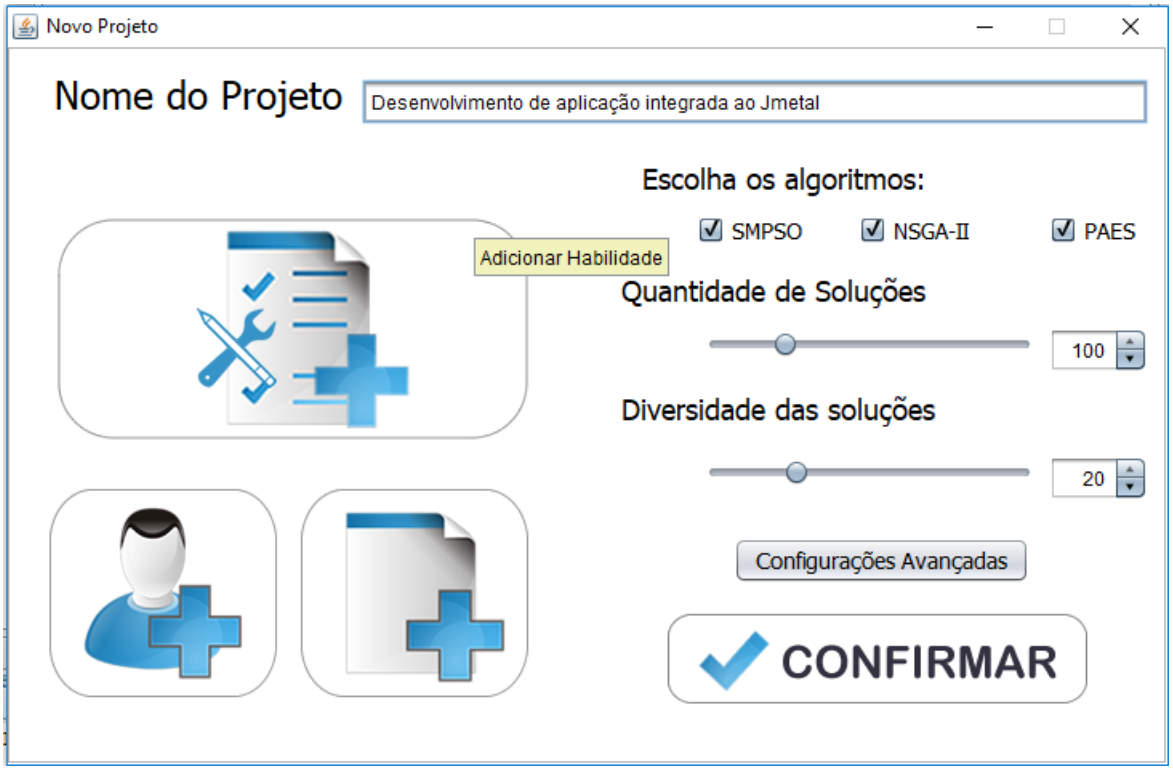
Quando o usuário opta por criar um projeto, ele é direcionado para a tela mostrada na Figura 18, provendo a aplicação com os dados: as habilidades necessárias para completar o projeto, as tarefas que compõem o projeto, e os funcionários disponíveis para realizar o projeto. Além disso, pode escolher quais algoritmos serão utilizados para gerar os possíveis escalonamentos para o projeto, e a quantidade de soluções e sua diversidade. É possível ainda ir na tela de configurações avançadas para alterar os parâmetros dos algoritmos, se achar adequado.

Figura 17 – Tela do usuário



Fonte: Próprio autor.

Figura 18 – Tela de criação de um novo projeto

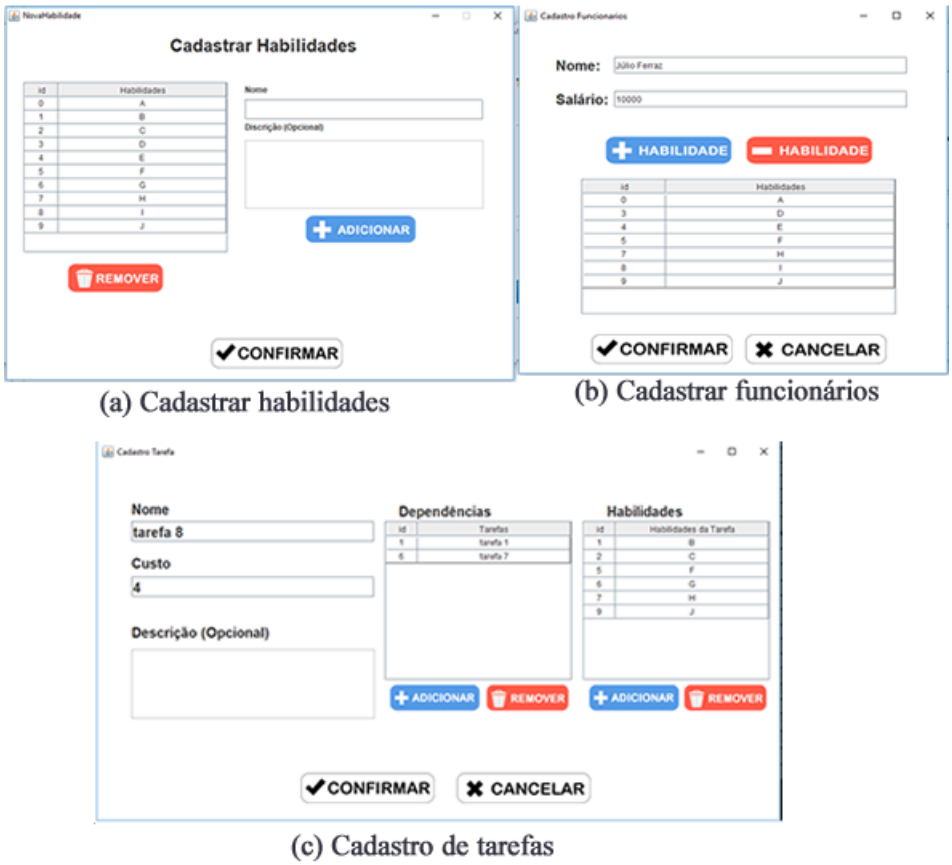


Fonte: Próprio autor.

Para realizar o cadastro dos funcionários, é necessário adicionar o nome, o salário e as habilidades que o mesmo possui. Já para adicionar as tarefas, é preciso adicionar o nome da tarefa, o custo, as tarefas que necessitam ser realizadas antes dela e as habilidades necessárias

para realizá-la. A Figura 19 apresenta as telas de cadastro de (a) habilidades, (b) funcionários e (c) tarefas.

Figura 19 – Telas de cadastro de habilidades, funcionários e tarefas



(a) Cadastrar habilidades

(b) Cadastrar funcionários

(c) Cadastro de tarefas

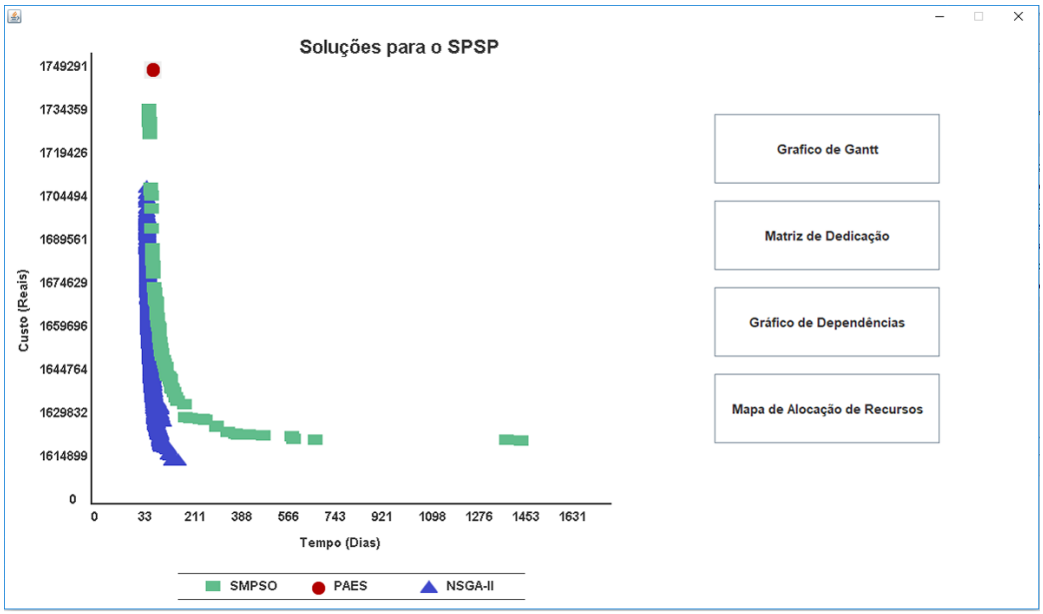
Fonte: Próprio autor.

Neste exemplo foram cadastradas dez habilidades necessárias para concluir o projeto. A partir das habilidades cadastradas foram cadastrados oito funcionários e dezesseis tarefas indicando qual das habilidades cadastradas ele possui ou requer para ser concluída. Caso falte alguma habilidade durante esses processos, basta que o gerente volte a tela de cadastro de habilidades para inserir a habilidade faltante.

Após o cadastro dos dados do projeto, o gerente volta para a tela usuário mostrada na figura 17 e pode iniciar a geração dos escalonamentos através do algoritmos, clicando no botão de execução. Ao término da execução dos algoritmos, os resultados gerados são exibidas em uma tela de soluções e dados do projeto Figura 20 . As soluções geradas pelo SMPSO, PAES e NSGA-II são representadas em um plano cartesiano onde o eixo x representa a escala de tempo e o y a de custo.

Para analisar uma solução específica ou dados do projeto basta clicar no objeto que representa a solução e clicar em um dos botões ao lado, que representam as diversas formas de visualização das soluções e dados do projeto.

Figura 20 – Tela de soluções e dados do projeto.



Fonte: Próprio autor.

Para decidir qual solução é mais adequada para o gerente, a IA-GPS disponibiliza recursos que permitem um olhar diferente sobre o projeto e sobre as soluções geradas. Os recursos para analisar as soluções ou dados do projeto são: o gráfico de Gantt, a matriz de dedicação, o gráfico de dependências e o mapa de alocação de recursos.

Na tela que mostra o gráfico de Gantt (Figura 21) o gerente pode ver, de acordo com o escalonamento gerado, quando cada tarefa começa e termina, quais tarefas serão realizadas antes, e quais serão realizadas em paralelo, além de ver sua duração. Para elaborar essa tela foi utilizado o *framework* gráfico jFreeChart [1].

A tela com a matriz de dedicação (Figura 22) fornece a relação entre tarefas e funcionários, mostrando a quantidade de dedicação que o funcionário dedica a uma tarefa específica no escalonamento encontrado e permite que essa solução seja exportada em formato xls e xml para poder ser vista em algum programa de planilha que desejar.

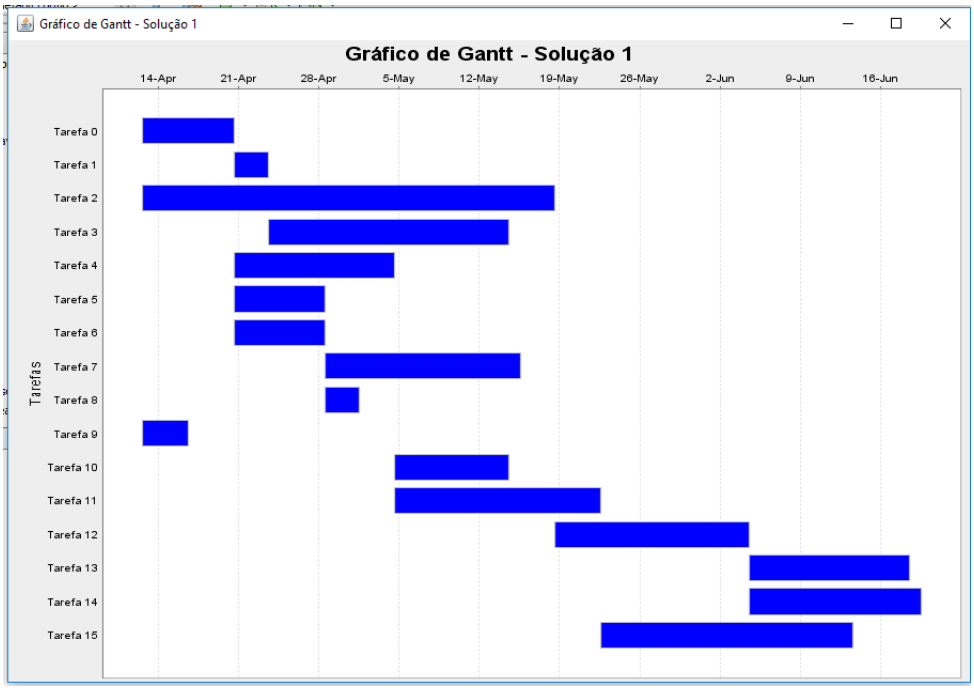
O mapa de alocação (Figura 23) de recursos mostra a porcentagem de dedicação de cada funcionário a uma tarefa para o escalonamento encontrado.

O grafo de pendências (Figura 29) mostra os relacionamentos entre tarefa e com ele vemos quais tarefas precisam ser concluídas antes das outras.

Como os algoritmos geram muitas soluções e essas às vezes são inválidas, então a aplicação faz a verificação de validade das soluções geradas. Verifica-se se suas funções objetivos possuem valores válidos, valores maiores que zero, e valores de dedicação de funcionários inferiores ou iguais a um.

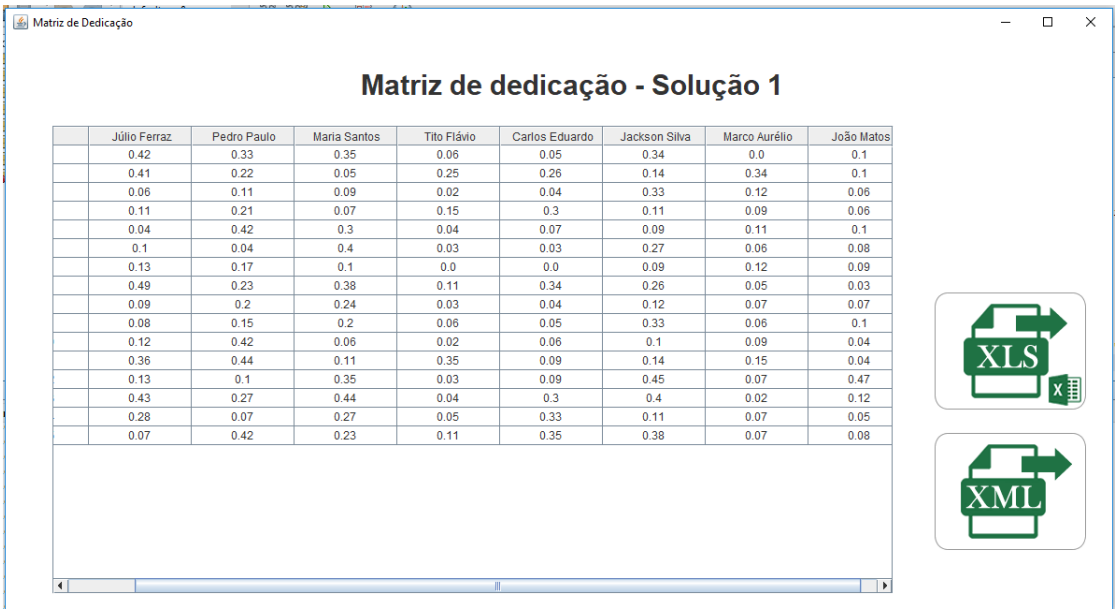
[1] Disponível em: <http://www.jfree.org/jfreechart/>.

Figura 21 – Tela gráfico de Gantt.



Fonte: Próprio autor

Figura 22 – Tela de matriz de dedicação

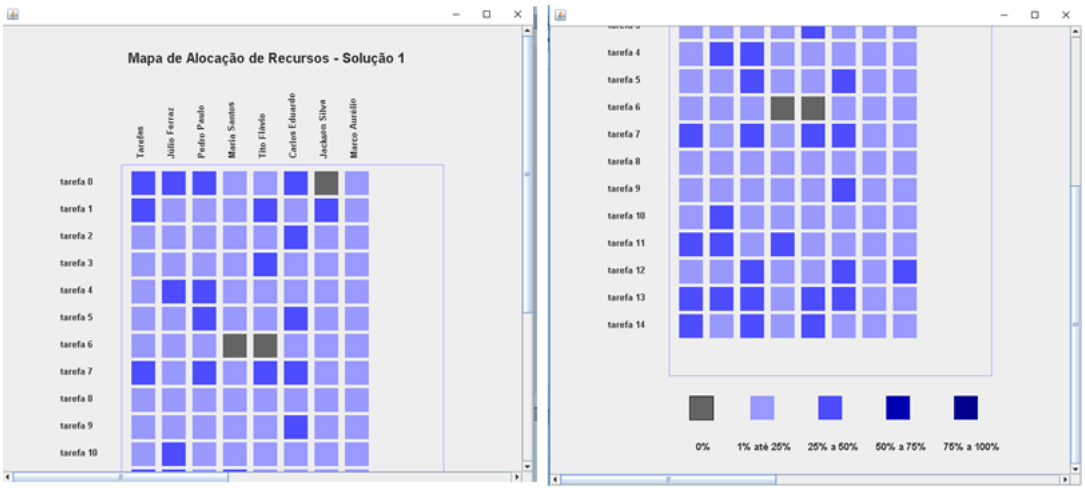


	Júlio Ferraz	Pedro Paulo	Maria Santos	Tito Flávio	Carlos Eduardo	Jackson Silva	Marco Aurélio	João Matos
	0.42	0.33	0.35	0.06	0.05	0.34	0.0	0.1
	0.41	0.22	0.05	0.25	0.26	0.14	0.34	0.1
	0.06	0.11	0.09	0.02	0.04	0.33	0.12	0.06
	0.11	0.21	0.07	0.15	0.3	0.11	0.09	0.06
	0.04	0.42	0.3	0.04	0.07	0.09	0.11	0.1
	0.1	0.04	0.4	0.03	0.03	0.27	0.06	0.08
	0.13	0.17	0.1	0.0	0.0	0.09	0.12	0.09
	0.49	0.23	0.38	0.11	0.34	0.26	0.05	0.03
	0.09	0.2	0.24	0.03	0.04	0.12	0.07	0.07
	0.08	0.15	0.2	0.06	0.05	0.33	0.06	0.1
	0.12	0.42	0.06	0.02	0.06	0.1	0.09	0.04
	0.36	0.44	0.11	0.35	0.09	0.14	0.15	0.04
	0.13	0.1	0.35	0.03	0.09	0.45	0.07	0.47
	0.43	0.27	0.44	0.04	0.3	0.4	0.02	0.12
	0.28	0.07	0.27	0.05	0.33	0.11	0.07	0.05
	0.07	0.42	0.23	0.11	0.35	0.38	0.07	0.08

Fonte: Próprio autor.

Além disso ,verifica se os dados de entrada do projeto são consistentes, ou seja, verifica se os funcionários cadastrados possuem as habilidades necessárias para executar por completo o projeto.

Figura 23 – Tela mapa de distribuição de dedicação

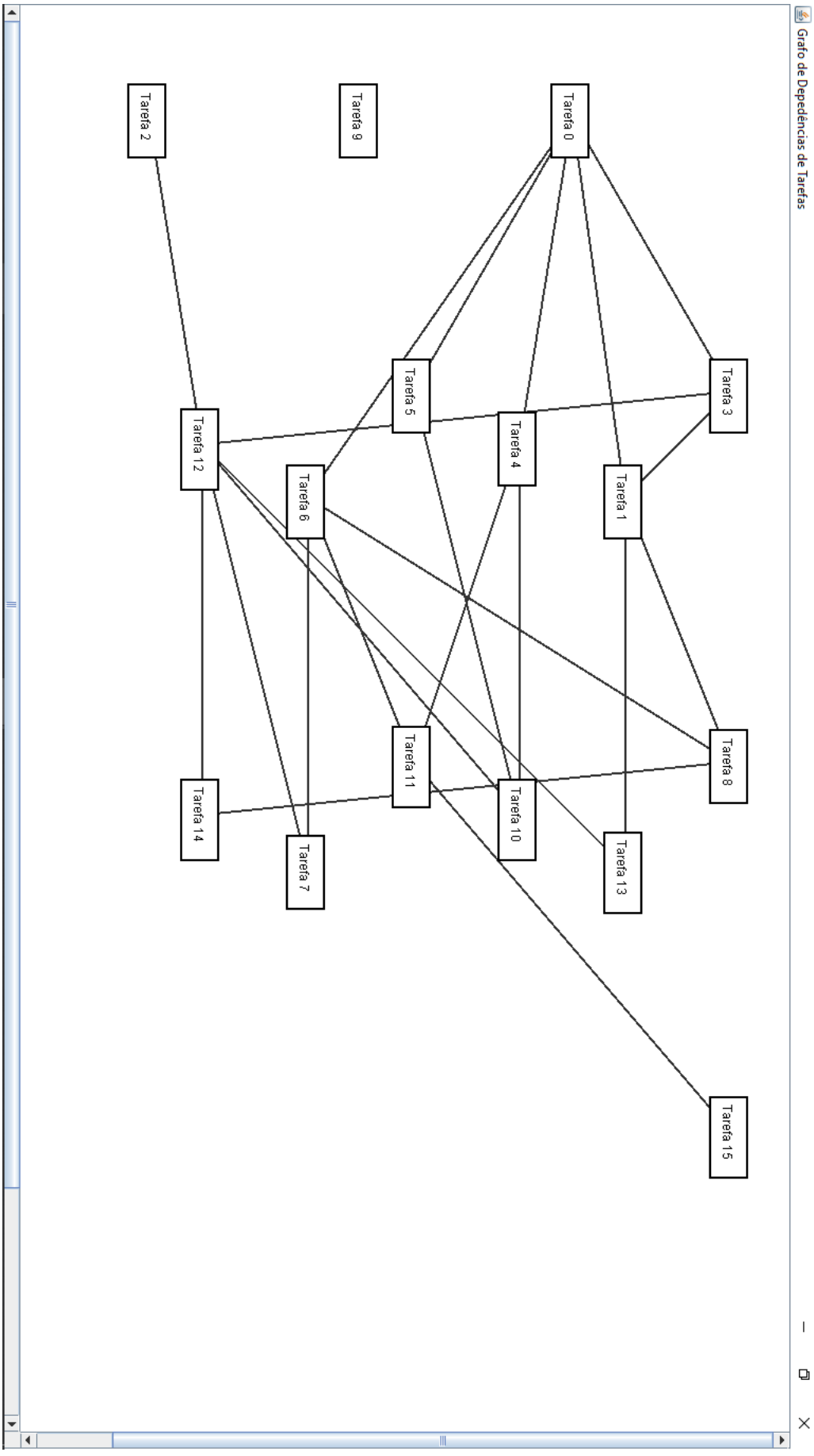


Fonte: Próprio autor

3.6 Validação da Ferramenta

A validação do IA-GPS foi feita pelo autor deste trabalho, de forma manual, utilizando dados sintéticos das instâncias do problema SPSP, propostas em (ALBA; CHICANO, 2007).

Figura 24 – Tela com grafo de dependência de tarefas



Fonte: Próprio autor.

4
Conclusões

O objetivo deste trabalho foi desenvolver uma aplicação que simplifica a utilização de algoritmos heurísticos, provendo uma interface mais amigável para os engenheiros de software e permitindo analisar as soluções geradas para o Problema do Escalonamento e Atribuição de Tarefas em Projetos de Software.

Para alcançar esses objetivos fizemos uma revisão na literatura sobre as necessidades de um gerente de projeto de software, projetamos e desenvolvemos uma aplicação que é integrada ao *framework* jMetal e que permite analisar os dados brutos contidos nas soluções geradas pelos algoritmos heurísticos, implementados no jMetal. Para facilitar a análise das soluções do SPSP, foram implementadas várias formas de visualização das soluções, como o gráfico de Gantt, a matriz de dedicação, o grafo de dependências das tarefas, o mapa de alocação de recursos e as soluções na fronteira de Pareto. A aplicação foi validada com dados de problemas extraídos da literatura.

O diferencial do IA-GPS em relação a ferramentas semelhantes, é sua integração com framework jMetal, permitindo a utilização de algoritmos, implementados neste framework. Além disso, através de uma interface gráfica, possibilita a escolha de soluções com base na fronteira de Pareto. A aplicação permite ainda as funções de importar e exportar projetos em formato xml de outros programas de gerenciamento de projetos, como o Microsoft Project, e exporta as soluções em xls para visualizá-las em aplicações de planilhas, como o Excel.

O IA-GPS possui algumas limitações, pois adota apenas o modelo estático de SPSP, de maneira que não é possível alterar o projeto criado, caso um funcionário ou uma tarefa sejam adicionados ou removidos durante o projeto. Caso ocorra uma modificação é necessário criar um novo projeto. Além disso, para estender a aplicação para considerar outros algoritmos implementados no jMetal é necessário fazer adaptações no código da aplicação.

Como trabalho futuro pretendemos estender a aplicação para permitir fazer escalonamentos dinâmicos, utilizando o DSPSP, que é a versão dinâmica do SPSP. Além disto, podemos

incorporar a utilização de técnicas de amostragem para reduzir a quantidade de soluções apresentadas, caso elas tenham desempenho parecidos.

Referências

ALBA, E.; CHICANO, J. F. Software project management with gas. *Information Sciences*, v. 177, n. 11, p. 2380 – 2401, 2007. ISSN 0020-0255. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0020025507000175>. Citado 5 vezes nas páginas 12, 13, 15, 17 e 42.

BEUME, N. et al. On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation*, v. 13, n. 5, p. 1075–1082, 2009. ISSN 1089-778X. Citado na página 21.

BIJU, T. A. A. V. A. C.; MOHANASUNDARAM, K. An improved differential evolution solution for software project scheduling problem. *The Scientific World Journal*, v. 2015, p. 9, 2015. Citado na página 18.

COELLO, C. C.; LAMONT, G.; VELDHUIZEN, D. van. *Evolutionary Algorithms for Solving Multi-Objective Problems*. 2nd. ed. Berlin, Heidelberg: Springer, 2007. (Genetic and Evolutionary Computation). Disponível em: <http://books.google.com/books?id=rXIuAMw3lGAC>. Citado na página 19.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Trans. Evol. Comp*, IEEE Press, Piscataway, NJ, USA, v. 6, n. 2, p. 182–197, 2002. ISSN 1089-778X. Disponível em: <http://dx.doi.org/10.1109/4235.996017>. Citado na página 20.

DEBLAERE, F.; DEMEULEMEESTER, E.; HERROELEN, W. Rescon: Educational project scheduling software. v. 19, p. 327 – 336, 2011. Citado 2 vezes nas páginas 12 e 27.

DURILLO, J. J.; NEBRO, A. J. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, v. 42, p. 760–771, 2011. ISSN 0965-9978. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0965997811001219>. Citado na página 27.

FERRUCCI, F.; HARMAN, M.; SARRO, F. Search-based software project management. In: *Software Project Management in a Changing World*. [S.l.]: Springer, 2014. p. 373–399. Citado na página 11.

FOWLER, M.; SCOTT, K. *UML essencial: um breve guia para a linguagem-padrão de modelagem de objetos*. Bookman, 2000. ISBN 9788573077292. Disponível em: <https://books.google.com.br/books?id=WDyxPgAACAAJ>. Citado na página 23.

GOLDBARG, E.; GOLDBARG, M.; LUNA, H. *Otimização Combinatória e Metaheurísticas: Algoritmos e Apliacações*. Elsevier Editora Ltda., 2017. ISBN 9788535278132. Disponível em: <https://books.google.com.br/books?id=STBbDwAAQBAJ>. Citado na página 19.

HARMAN, M.; JONES, B. F. Search-based software engineering. *Information and Software Technology*, v. 43, n. 14, p. 833 – 839, 2001. ISSN 0950-5849. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0950584901001896>. Citado na página 20.

HARMAN, M.; MANSOURI, S. A.; ZHANG, Y. Search based software engineering : A comprehensive analysis and review of trends techniques and applications. In: . [S.l.: s.n.], 2009. Citado na página 11.

JACOBSON, I. et al. *Object-Oriented Software Engineering*. [S.l.]: Addison-Wesley, 1993. Citado na página 23.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. [S.l.: s.n.], 1995. v. 4, p. 1942–1948 vol.4. Citado na página 20.

KNOWLES, J.; CORNE, D. The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In: *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. [S.l.: s.n.], 1999. v. 1, p. 98–105 Vol. 1. Citado na página 20.

KUPP, N.; MAKRIS, Y. Applying the model-view-controller paradigm to adaptive test. *IEEE Design Test of Computers - IEEE DToC*, v. 29, p. 28–35, 2012. Citado na página 26.

MARTINS, P.; LAUGENI, F. *Administração da produção*. Saraiva, 2005. ISBN 9788502046160. Disponível em: <https://books.google.com.br/books?id=fYfuOwAACAAJ>. Citado na página 11.

PMI. *Um Guia Do Conhecimento Em Gerenciamento de Projetos (guia PMBOK): [Brazilian Portuguese Version of: a Guide to the Project Management Body of Knowledge (PMBOK Guide)*. Project Management Institute, 2014. (Pmbok® Guide). ISBN 9781628250077. Disponível em: <https://books.google.com.br/books?id=BfJGnwEACAAJ>. Citado na página 11.

PRESSMAN, R. *Engenharia de Software - 7.ed.* McGraw Hill Brasil, 2011. ISBN 9788580550443. Disponível em: <https://books.google.com.br/books?id=y0rH9wuXe68C>. Citado 3 vezes nas páginas 23, 24 e 26.

REENSKAUG, T. M. H. *Thing-Model-View-Editor – an Example from a planningsystem*. 1979. Http://heim.ifi.uio.no/ trygver/1979/mvc-1/1979-05-MVC.pdf. Erste Notiz zum Modell-View-Controller-Paradigma mit exemplarischen Beispielen verfasst vom Erfinder persönlich. Disponível em: <http://heim.ifi.uio.no/~trygver/1979/mvc-1/1979-05-MVC.pdf>. Citado na página 26.

SALAS-MORERA, L. et al. Ppcproject: An educational tool for software project management. *Computers Education*, v. 69, p. 181 – 188, 2013. ISSN 0360-1315. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0360131513001851>. Citado 2 vezes nas páginas 12 e 28.

SOMMERVILLE, I. *Engenharia de software*. PEARSON BRASIL, 2011. ISBN 9788579361081. Disponível em: <https://books.google.com.br/books?id=H4u5ygAACAAJ>. Citado 4 vezes nas páginas 22, 25, 29 e 30.

STANDISH GROUP. Chaos report. 2014. Citado na página 11.

STYLIANOU, C.; GERASIMOU, S.; ANDREOU, A. S. A novel prototype tool for intelligent software project scheduling and staffing enhanced with personality factors. In: *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*. [S.l.: s.n.], 2012. v. 1, p. 277–284. ISSN 1082-3409. Citado 2 vezes nas páginas 12 e 28.

XIAO, J.; AO, X.-T.; TANG, Y. Solving software project scheduling problems with ant colony optimization. *Computers Operations Research*, v. 40, n. 1, p. 33 – 46, 2013. ISSN 0305-0548. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0305054812001104>. Citado 2 vezes nas páginas 12 e 13.

XIAO, J.; GAO, M.-L.; HUANG, M.-M. Empirical study of multi-objective ant colony optimization to software project scheduling problems. In: *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2015. (GECCO '15), p. 759–766. ISBN 978-1-4503-3472-3. Disponível em: <http://doi.acm.org/10.1145/2739480.2754702>. Citado 2 vezes nas páginas 12 e 13.

Apêndices

APÊNDICE A – Casos de uso comentados

Neste apêndice apresentamos o detalhamento dos principais casos de uso da ferramenta IA-GPS.

Figura 25 – Descrição de caso de uso - Cadastrar os dados do Projeto

Caso de Uso (CSU4)	Cadastrar os dados do projeto.
Objetivo:	Informar componentes necessárias para concluir um projeto.
Requisitos Funcionais:	RF02.
Prioridade:	Alta.
Frequência de uso:	Alta.
Pré-condições:	1. Ter feito login na ferramenta.
Pós-condições:	Dados da tarefa salvo no sistema.
Fluxo Principal:	1. Cadastrar as habilidades necessárias para conclusão do projeto. 2. Cadastrar tarefa. 3. Cadastrar funcionários.
Fluxo Alternativo:	-
Fluxo Exceção:	E1 - Cadastro de Tarefa, habilidade ou funcionário já existente. 1. Mensagem de erro. E2 – Custo da tarefa e/ou salário do funcionário menor ou igual a zero. 1. Mensagem de erro. E3 - Habilidade ou tarefa necessária não cadastrada. 1. Cadastrar tarefa ou habilidade antes de cadastrar nova tarefa. E4 – funcionários não possuem habilidades o suficiente para concluir o projeto. 1. Cadastrar funcionário ou adicionar uma habilidade a funcionário já cadastrado.

Fonte: Próprio autor.

Figura 26 – Descrição de caso de uso - Visualizar o diagrama de Gantt.

Caso de Uso (CSU10)	Visualizar o diagrama de Gantt.
Objetivo:	Visualizar as tarefas ao longo do tempo.
Requisitos Funcionais:	RF06.
Prioridade:	Média.
Frequência de uso:	Alta.
Pré-condições:	1. Ter feito login na ferramenta. 2. Ter cadastrado as habilidades, funcionários e tarefas do projeto. 3. Ter executado o(s) algoritmo(s).
Pós-condições:	-
Fluxo Principal:	1. Após a execução do(s) algoritmo(s), o gerente escolhe uma das soluções da fronteira de Pareto. 2. Com a solução selecionada, o gerente escolhe a função de visualizar o gráfico de Gantt.
Fluxo Alternativo:	-
Fluxo Exceção:	E1 – Nenhuma solução selecionada. 1 – Mensagem de erro.

Fonte: Próprio autor.

Figura 27 – Descrição de caso de uso - Visualizar o grafo de dependências.

Caso de Uso (CSU11)	Visualizar o grafo de dependências.
Objetivo:	Visualizar quais tarefas necessitam serem solucionadas primeiro em um projeto.
Requisitos Funcionais:	RF07.
Prioridade:	Média.
Frequência de uso:	Alta.
Pré-condições:	1. Ter feito login na ferramenta. 2. Ter cadastrado as habilidades e tarefas do projeto. 3. Ter executado o(s) algoritmo(s).
Pós-condições:	-
Fluxo Principal:	1. Após a execução do(s) algoritmo(s), o gerente escolhe uma das soluções da fronteira de Pareto. 2. Com a solução selecionada, o gerente escolhe a função de visualizar o grafo de dependências.
Fluxo Alternativo:	-
Fluxo Exceção:	E1 – Nenhuma solução selecionada. 1 – Mensagem de erro.

Fonte: Próprio autor.

Figura 28 – Descrição de caso de uso - Visualizar a matriz de dedicação.

Caso de Uso (CSU12)	Visualizar matriz de dedicação.
Objetivo:	Visualizar qual funcionário foi escalonado para cada tarefas e quanto de sua dedicação ele irá se dedicar a ela.
Requisitos Funcionais:	RF09.
Prioridade:	Alta.
Frequência de uso:	Alta.
Pré-condições:	1. Ter feito login na ferramenta. 2. Ter cadastrado as habilidades, tarefas e funcionários do projeto. 3. Ter executado o(s) algoritmo(s).
Pós-condições:	-
Fluxo Principal:	1. Após a execução do(s) algoritmo(s), o gerente escolhe uma das soluções da fronteira de Pareto. 2. Com a solução selecionada, o gerente escolhe a função de visualizar a matriz de dedicação.
Fluxo Alternativo:	-
Fluxo Exceção:	E1 – Nenhuma solução selecionada. 1. – Mensagem de erro.

Fonte: Próprio autor.

Figura 29 – Descrição de caso de uso - Visualizar o mapa de alocação de recursos.

Caso de Uso (CSU13)	Visualizar o mapa de alocação de recursos do projeto.
Objetivo:	Visualizar quais tarefas cada funcionário se dedica por mais tempo.
Requisitos Funcionais:	RF08.
Prioridade:	Média.
Frequência de uso:	Alta.
Pré-condições:	1. Ter feito login na ferramenta. 2. Ter cadastrado as habilidades, tarefas e funcionários do projeto. 3. Ter executado o(s) algoritmo(s).
Pós-condições:	-
Fluxo Principal:	1. Após a execução do(s) algoritmo(s), o gerente escolhe uma das soluções da fronteira de Pareto. 2. Com a solução selecionada, o gerente escolhe a função de visualizar o mapa de alocação de recursos.
Fluxo Alternativo:	-
Fluxo Exceção:	E1 – Nenhuma solução selecionada. 1. – Mensagem de erro.

Fonte: Próprio autor.